Australian Government
Department of Defence
Defence Science and Technology Organisation

A Proof-of-concept Study on Integrating GEANT4 and MATLAB to Develop a Simulation Capability for Testing Radiological Source Localisation Algorithms

Alaster Meehan, Ajith Gunatilaka, Damian Marinaro and Michael Roberts

Human Protection and Performance Division
Defence Science and Technology Organisation

DSTO-TN-0995

ABSTRACT

This report describes a proof-of-concept study on integrating a Monte Carlo particle simulation package called GEANT4 with MATLAB technical computing software to build a radiological simulation capability to test and evaluate radiological source localisation algorithms developed in MATLAB. The purpose of this report is to document what was learnt during this project, including alternative approaches that were considered; useful instructions and tips; and bugs and pitfalls, so that these can benefit anyone who is undertaking to extend this work.

Published by

Human Protection and Performance Division
DSTO Defence Science and Technology Organisation
506 Lorimer St
Fishermans Bend, Victoria 3207 Australia

Telephone: (03) 9626 7000
Fax: (03) 9626 7999

AR-014-956
February 2011

A Proof-of-concept Study on Integrating GEANT4 and MATLAB to Develop a Simulation Capability for Testing Radiological Source Localisation Algorithms

Executive Summary

A capability for rapid detection and localisation of radiological sources can be of great benefit to first responders during a radiological threat scenario. For example, the early detection and localisation of radiological material in a radiological dispersal device (a "dirty bomb") before it is detonated could prevent the dispersal of radioactive particles into the atmosphere and the consequent radiation exposure to people. Therefore, there is significant research interest in developing radiological source backtracking algorithms that can enable more efficient search methods to reduce search times. Conducting radiological field trials to test and evaluate these algorithms is laborious and time consuming because of the large amounts of data required and also involves the inherent hazards of working with radioactive sources. Therefore, developing a capability that enables testing of search algorithms in a realistic simulation environment is considered important. This report describes a "proof-of-concept" study carried out to explore the possibility of developing such a capability.

This study involved the integration of the Monte Carlo particle simulation package called GEANT4 with the MATLAB scientific programming software. GEANT4 is a powerful toolkit that is capable of accurately modelling radioactive particles and their interactions with matter. The present study used GEANT4 to simulate radiation sources, a radiation detector, and an environment consisting of simple obstacles. By providing an interface to communicate between GEANT4 and MATLAB, a radiological source search algorithm running in MATLAB was allowed to successfully carry out source backtracking in the simulated environment.

This report describes the technical details of the software integration, including how to set up GEANT4. Alternative options considered for communication between GEANT4 and MATLAB, problems encountered and solutions or workarounds found, and general lessons learnt are also discussed in the report.

We recommend conducting further work to extend the limited capability that was achieved in the study described in this report. For example, adding a graphical user interface to more conveniently define obstacles such as buildings in the simulated environment would make this tool more user-friendly and easy to use.

Contents

1. Introduction

Due to the threat of radiological attacks by terrorists on defence forces and civilians, research on detection and localisation of radiological materials has received significant interest. The Human Protection and Performance Division (HPPD) of Defence Science and Technology Organisation (DSTO) has been involved in the test and evaluation of radiation detectors and also in developing radiological source localisation algorithms. While HPPD scientists have carried out some field trials in the past to test radiation survey equipment and acquire experimental data to test and validate source localisation algorithms, they have increasingly recognised the importance and benefits of building a radiological modelling and simulation capability within HPPD to complement experimentation. Compared to radiological experimentation in the lab or the field, radiological modelling and simulation is less costly in terms of both time and resources, more flexible, and avoids the hazards of working with ionising radiation. Where field experimentation is needed, simulation can also be used to design more effective and efficient field trials.

To establish the desired radiological modelling and simulation capability, we plan on using the software package GEANT4 (GEometry ANd Tracking) which is a C++ based software toolkit, originally developed by CERN, http://geant4.cern.ch/, for the simulation of the passage of particles through matter[1]. GEANT4 provides comprehensive detector and physics modelling capabilities within a flexible structure that allows significant control over a range of functionalities such as geometry and material specification, particle tracking, interaction events, detector response, and visualisation. Importantly, GEANT4 is correctly termed a "toolkit" in that a user may assemble their program from components chosen from the supplied code or specifically built for their application. We intend to use the GEANT4 package for a variety of tasks, including simulating and analysing new personal dosimetry device designs [2, 3], testing radiological source search algorithms and evaluating standoff and novel radiation detection systems.

While one or more searchers equipped with gamma radiation detectors may search for a putative radiological source or sources in an area using a uniform scan or even a random search pattern, there is a need to develop more efficient search algorithms that help reduce the search time and radiation exposure. We have developed both batch [4, 5] and sequential [6, 7] approaches for radiological source localisation. Sequential approaches are often preferred because with these algorithms there is no need to reprocess previously processed data when new data are acquired.

This report describes a small "proof of concept" study carried out to demonstrate the feasibility of using GEANT4 to test a sequential radiological source search algorithm [6]. In this study, the GEANT4 package was used to simulate a source(s) of radiation, track the interaction of the emitted radiation within a complex environment, and simulate the response of a generic radiation detector in terms of the radiation counts measured at a specified position within the environment. These simulated detector measurements were then used as input to the radiological source search algorithm. The search algorithm then used these simulated measurements to update source estimates and determine the optimum detector location for acquiring the next measurement to maximise information gain. The control vector describing the next detector position was passed back to GEANT4 so that the simulated detector could be moved to begin the next data acquisition cycle.

Because our search algorithm development work is being carried out in MATLAB, a major component of this study was to investigate various methods for communication between MATLAB and GEANT4 and to thereby implement the most optimal or flexible approach whilst taking into consideration the time costs involved in implementing a particular method in future studies. The main effort of this project was therefore to develop an effective and robust method of sending data between GEANT4 and MATLAB, and is described in Section 2. Once accomplished, the effectiveness of a selected search algorithm in realistic simulated environments could be tested and a proof of concept example is described in Section 3. Throughout this report, the lessons learnt during the project will be documented, including alternative approaches that were considered, useful instructions and tips, as well as bugs and pitfalls encountered, so that these can benefit anyone who is undertaking to extend this work.

2. Technical Methods

2.1 Communications between MATLAB and the GEANT4 server

2.1.1 Methods of communication between MATLAB and GEANT4

There were several methods that were considered for communication between MATLAB and GEANT4, each with its own advantages and disadvantages. One of the primary limitations to establishing communications between the software packages is that GEANT4 was principally written to run on a Linux-based machine, Matlab users where operating in a windows based environment. GEANT4 may be run on a Windows system but has less support and the potential for more bugs. The recommended installation process for installing GEANT4 on a windows machine requires the Cygwin a Linux emulator.

There was also some contention wether to have GEANT4 running on a local machine or a server. Maintaining a local copy of GEANT4 on a user's machine is relatively straightforward and can harness the processing power that is generally unused on a desk-top machine. However, since GEANT4 and the physics data files are regularly patched and upgraded, having a common installation located on an accessible networked server allows for simpler maintenance and consistency of version to be ensured.

After running GEANT4 on a local Windows machine, it was found that there were several instabilities that caused GEANT4 to crash, including the radioactive decay module, which were not evident when testing the same program on a Linux machine. A consequence of the greater use of Linux-based systems in the High Energy Physics community is that there is less support for GEANT4 operation in the Windows environment. Hence it was decided to run GEANT4 on Linux. This limited the communication between Matlab and GEANT4 to methods that could work under Windows and Linux and have the ability to communicate between different machines.

The following methods for communication between MATLAB and GEANT4 were considered.

2.1.1.1 The MATLAB system() function

MATLAB can call executable files by using the **system ()** function. Through the system () function MATLAB can open an instance of GEANT4 and pass a macro file that can specify various commands to GEANT4, see section 2.4 below. The disadvantage of this method is that every time MATLAB makes a command to GEANT4, a new instance of GEANT4 is created, and hence there is a lot of overhead in computing time as all of the initialisation processes are repeated.

2.1.1.2 Windows COM commands

The Component Object Model (COM) is an application programming interface developed by Microsoft that enables software components to communicate in a Windows environment. MATLAB has built in functions that enable COM objects to be easily accessed. GEANT4 can use C++ programming mechanisms that simplify the implementation of COM objects. Using COM functionality it is possible to concurrently communicate between MATLAB and GEANT4. The main disadvantage of this method is that it is only compatible with Windows programs and would not support communication with a Linux based server, thereby ruling this method out.

2.1.1.3 Piping commands from MATLAB

Piping is where the standard output of one process is sent to the standard input of another. There is an open source version of the popen() function for MATLAB available at:
http://www.mathworks.com/matlabcentral/fileexchange/13851

Being able to pipe input to GEANT4 would reduce the overhead in sending commands, as a new instance of GEANT4 would not need to be created each time. However this method only supports one way communication and the small amount of overhead this method would save would not be worth the time and effort needed in getting this method to work as required. Therefore this method has not been implemented in this project.

2.1.1.4 Using Sockets

Information and open source software for using sockets – endpoints for communication between two machines – in MATLAB are available at:
http://iheartmatlab.blogspot.com/2008/08/tcpip-socket-communications-in-matlab.html

This is likely to be an optimal method for communicating with the Linux server through MATLAB, but would require a significant amount of time to develop the correct code. This method was considered but not implemented in this project due to time constraints; passing files through the MATLAB system command as described in Section 2.3.1.1 is likely to be more flexible and required less time to develop a working solution, while the benefits in reducing the computing overhead through the use of sockets would be minimal.

2.1.1.5 MEX files

This would involve writing MEX (MATLAB EXecutable) functions, which are subroutines produced from source code that can be run from within MATLAB, that MATLAB could call to run GEANT4 as required and pass the necessary information. While this method would be very fast and powerful, it is likely that it would only work if GEANT4 was running on the local machine and may not address the computing overhead.

2.1.1.6 Writing the search algorithms in C++

Transferring the search algorithm code into C++ for direct implementation within the GEANT4 environment would offer the fastest method for testing search algorithms within GEANT4. However, the continuing development of the search algorithms is occurring under MATLAB, so it is not yet appropriate to convert the MATLAB code directly into C++.

2.1.2 Using the MATLAB system() function

The MATLAB system() function was the method that was ultimately implemented after considering the alternatives described above. It was found that the overhead in calling GEANT4 and passing files to and from the server was only 3–4 seconds, compared to the runtime which is usually 30–40 seconds or far longer (Fig.1); therefore this overhead was not considered to be significant.

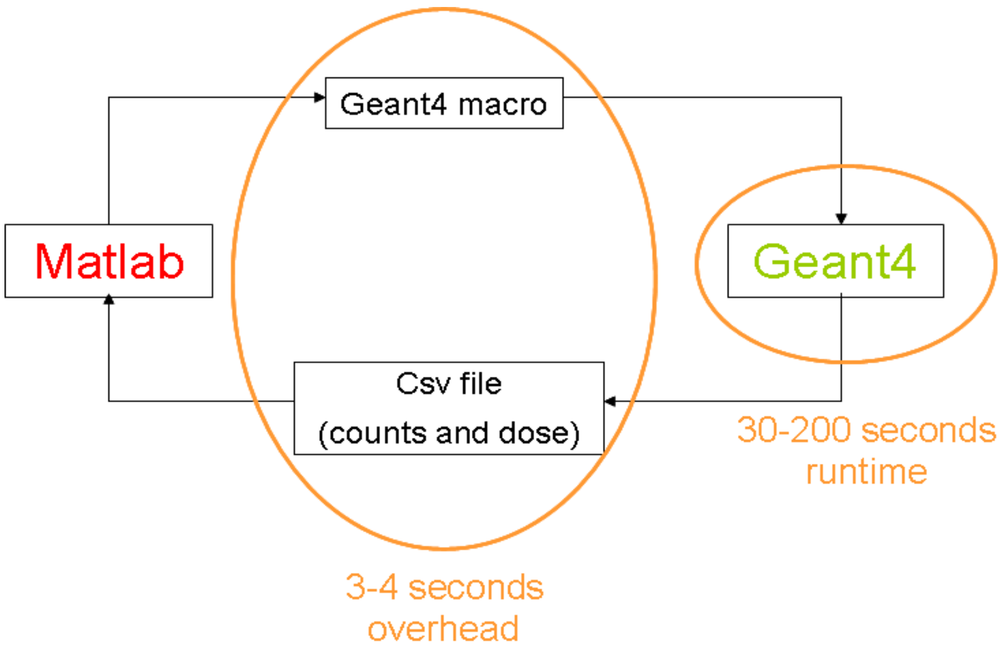

Figure 1: Communication between MATLAB and GEANT4

The simulation is fully controlled through the MATLAB interface by the following process:

1) MATLAB starts
2) MATLAB code writes out a GEANT4 macro file
3) MATLAB invokes GEANT4, passing the name of the macro file as a command line argument
4) GEANT4 runs a simulation
5) GEANT4 writes out a results file
6) MATLAB waits for GEANT4 to finish processing
7) MATLAB reads the GEANT4 results file
8) MATLAB modifies the current simulation parameters, going back to step 2. if required.

MATLAB uses the Linux "ssh" and "scp" commands in the function "GetCountsG4.m" to communicate with the GEANT4 Linux Server. For instructions in using these commands in Microsoft Windows see section 2.1.2.

2.1.3 Using ssh Commands in Windows

The method chosen for MATLAB and GEANT4 communication involves MATLAB calling executable files by using the **system()** function, as described in Section 2.3.1.1. To establish the connection, a MATLAB routine run on a local machine can use the Linux commands **ssh** and **scp** to communicate with the Linux GEANT4 server. These commands can be accessed by installing Cygwin from the http://www.cygwin.com/ website and ensuring that the **ssh** components are added to the installation.

MATLAB then uses the system command to communicate with the server rather than passing a command to a Cygwin terminal. This is more direct and efficient but requires a few settings be established. First the Cygwin\bin directory needs to be in the Windows path environment variable.

1. Right click **my computer** then select **Properties**.
2. From the **Advanced** tab select **Environment Variables**.
3. Under system variables select **Path** then edit.
4. Append the **Variable value** to include the Cygwin drive and path e.g.**:**
   C:\cygwin\bin

To avoid being asked for a password every time, MATLAB can make an **ssh** or **scp** command public key authorisation between your machine and the GEANT4 server, as described in the next section.

2.1.4 Procedure for Setting up Public Keys for the GEANT4 server

Note: in the following, **username** refers to your user name on your PC and **accountname** refers to the account name you use on the GEANT4 server. The **geantserver** refers to the IP of the server running Geant.

From a **Cygwin terminal** or a **command prompt** (recommended) type

```
ssh-keygen -t rsa
```

Press Enter to use the default directory for saving public keys, and leave the pass phrase blank.

You should now have two keys, your private key "**id_rsa**" and your public key "**id_rsa.pub**" saved in either "**\cygwin\.ssh**" or "**cygwin\home\username\.ssh**".

When setting the public keys, ensure that the private key "**id_rsa**" is in the **cygwin\.ssh** and the **cygwin\home\username\.ssh** so you can ssh without a password from both Cygwin and the command prompt.

From here, copy the public key to the Linux server.

If using a cmd prompt then type on a single line:

```
scp C:\cygwin\.ssh\id_rsa.pub accountname@ geantserver:/home/linux/.ssh
```

From a Cygwin prompt type on a single line:

```
scp cygdrive/c/cygwin/home/username/.ssh/id_rsa.pub  accountname@
geantserver:/home/linux/.ssh
```

Log on to the server using:

```
ssh  accountname@ geantserver
```

You will be asked to enter a password.

Go to the .ssh directory.

```
cd .ssh
```

From your .ssh directory, append the authorised keys file with the following command:

```
cat id_rsa.pub >>authorized_keys
```

Remove your public key.

```
rm id_rsa.pub
```

Exit from the server by typing:

```
exit
```

Try logging on using ssh again using:

```
ssh  accountname@ geantserver
```

You should not have been asked for a password this time. If so, then you have finished setting up the public keys.

2.1.4.1 Troubleshooting

If you are asked for a password, then it is likely you need to set private permissions for the authorized_keys file.
Run these commands after logging onto the Linux server.

```
chmod 0600 home/accountname/.ssh/authorized_keys
```

permissions may also need to be set for the .ssh directory and your home directory.

```
chmod 0600 home/accountname/.ssh
chmod 0600 home/accountname
```

You may also have to set private permissions for your private keys if you are getting error messages. From Command prompt.

```
chmod 0600 c:\cygwin\.ssh\id_rsa
chmod 0600 c:\ cygwin\home\username\.ssh\id_rsa
```

Or from Cygwin type:

```
chmod 0600 cygdrive/c/cygwin/.ssh/id_rsa
chmod 0600 cygdrive/c/cygwin/home/username/.ssh/id_rsa
```

2.1.5 Calling GEANT4 using Shell Scripts and Batch Files

GEANT4 requires certain environment variables to be set before running. Because of this, MATLAB will actually call a shell script instead of directly calling GEANT4. All this shell script does is source the environment variables and then call GEANT4 with an option for a macro file to be passed to it. This is the shell script used:

run_test_bed.sh
```
source .bash_profile
cd $G4WORKDIR
bin/Linux-g++/test_bed_matlab $1
```

To call this shell script from MATLAB, you would use the following command.

```
System ('ssh linux@geantmg14.dsto.defence.gov.au "bash run_test_bed.sh macro.mac"');
```

where “macro.mac” is a GEANT4 macro that is used to send several commands to GEANT4 (see Section 2.4). Similar calls are made in the MATLAB file “GetCountsG4.m”, (see appendix A.2.2).

When running GEANT4 in a Windows environment, MATLAB should call a batch file that sets the environment variables and then runs GEANT4.

2.2 Control of GEANT4 Simulations Using Macro Files

GEANT4 has various built-in user interface commands to control almost every aspect of the simulation. These commands can be used interactively via the (Graphical) User Interface; they can be written in a script or macro file where each given command is executed in sequence or they can be called directly within the C++ code. The command-line interface is displayed when running a GEANT4 program in User Interface mode, such as in “**test_bed_matlab**” for example. Typing “**help**” at this interface brings up a list of available commands, as shown below:

```
Idle> help
Command directory path : /
Sub-directories :
 1) /control/   UI control commands.
 2) /units/   Available units.
 3) /persistency/   Control commands for Persistency package
 4) /geometry/   Geometry control commands.
 5) /tracking/   TrackingManager and SteppingManager control commands.
 6) /event/   EventManager control commands.
 7) /cuts/   Commands for G4VUserPhysicsList.
 8) /run/   Run control commands.
 9) /random/   Random number status control commands.
 10) /particle/   Particle control commands.
 11) /process/   Process Table control commands.
 12) /USER/   UI commands for the dosimeter converter
 13) /material/   Commands for materials
```

```
14) /physics_engine/   commands related to the physics simulation engine.
15) /gps/   General Paricle Source control commands.
16) /hits/   Sensitive detectors and Hits
17) /grdm/   Controls for the Radioactive Decay Module.
18) /vis/   Visualization commands.
Commands :

Type the number ( 0:end, -n:n level back ) :
```

Most of these commands are the in-built GEANT4 commands and more information on these in-built commands can be found in the GEANT4 manual. It is also possible for the user to define a list of specific commands. These are defined within the GEANT4 code using a 'G4UImessenger' base class. In this project, a set of user-defined commands are located under the directory **/USER**.

A macro file can be executed by using the command "**/control/execute test.mac**" where "**test.mac**" is a text file containing a list of GEANT4 commands. A macro file can also be passed to the executable, for example by calling "**test_bed_matlab test.mac**", at which point the executable will be run, macro file will be executed and, at completion, GEANT4 will be closed. Macro files can also contain the command **/control/execute** to execute other macros; hence with a variety of short macro files it is possible to control much of GEANT4's functionality. Therefore by passing macro files to an executable, much of GEANT4's functionality can be specified and controlled by programs such as MATLAB.

2.2.1 Setting the Detector Position

For generality while testing the GEANT4 – MATLAB communication and the search algorithm, the "detector" is modelled with a sphere of water 25cm in diameter. This generic detector model will be substituted with a realistic detector model, such as of a NaI(Tl) scintillator or a High-Purity Germanium crystal, in future studies depending on the requirements of the study. Within the generic detector, counts record the number of events that deposit energy within the detector sphere.

For the purposes of simulating a search routine, it is necessary to control the placement of the simulated radiation detector within the environment. The position of the detector can easily be set with the following user-defined command:

/USER/DetectorPosition ***x y z unit***

where "x" "y" "z" describe the coordinates of the centre of the detector position, and "unit" can be given in "m", "cm" "mm" etc; if "unit" is left blank, the default unit is "m". Note that, although there are no vertical or horizontal directions defined intrinsically in GEANT4, most graphical visualisations will orient the axes as shown in Figure 2, with the "y" ordinate as the vertical component.

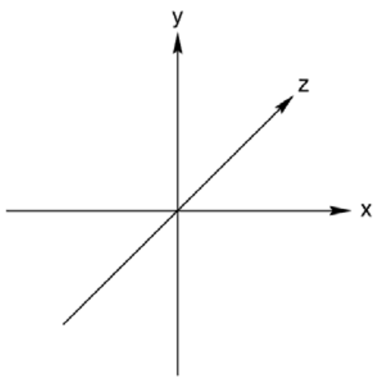

Figure 2: Coordinate axes as displayed by graphical visualisation of GEANT4 geometry

Further, since the MATLAB model only operates in 2D, the vertical position "y" is always set to 1m above the ground when MATLAB calls this command to move the detector in "GetCountsG4.m".

2.2.2 Constructing Environments

For this test bed the world size is a cube that is 200m by 200m by 200m with the origin at the centre. The bottom half of this world is filled with concrete to simulate the ground. These properties are hard coded in "**DetectorConstruction.cc**".

GEANT4 has a lot of functionality for describing complex shapes through various different C++ classes described in Chapter 4 of the GEANT4 User Guide. For this test bed, basic user commands have been developed to describe 3-dimensional blocks, where the size, position and material can be specified through the following commands:

/USER/Block/Size ***x y z unit***
where "*x*" "*y*" "*z*" specify the half-size of the block for each dimension along the coordinate axes as described in Figure 2 above and "unit" can be given in "m", "cm" "mm" etc, "m" being the default. Note that, due to the way in which GEANT4 defines geometrical dimensions, the full length of the block sides is twice that specified by the *x y z* values.

/USER/Block/Material ***name***
where "*name*" can be any material that is specified in the GEANT4 Material Database (see Appendix 8 of the GEANT4 user guide).

/USER/Block/Position ***x y z unit***
where "*x*" "*y*" "*z*" specifies the geometrical centre of the block to be created and "*unit*" is defined in the same way as above.

/USER/Block/Build
This command does not take any arguments but will add a block to the existing environment with the previously specified properties. A block with size (0, 0, 0) will generate a warning message and will not be constructed. If a block property has not been previously defined, default values are used; the default position is (0, 0, 0) and the default

material is G4_CONCRETE. It should be noted that the default block size is defined as (0, 0, 0), but as described above this will generate a warning and the block will not be constructed.

/USER/Block/RemoveAll
Removes all blocks from the environment.

These commands can be written into a macro file to describe the environment. It is possible to specify the size and material of a block and then specify multiple positions to build blocks of the same type. The following macro would build three blocks that are 1m^3 aligned in a row on the ground:

```
/USER/Block/Size 0.5 0.5 0.5
/USER/Block/Material G4_CONCRETE

/USER/Block/Position -2 0.5 0
/USER/Block/Build

/USER/Block/Position 0 0.5 0
/USER/Block/Build

/USER/Block/Position 2 0.5 0
/USER/Block/Build
```

The following macro uses different shapes and materials to construct the scene that is illustrated in Figure 3. Note that all blocks appear in the default colour of yellow; functionality can be added at a later date to specify colours for better visualisation.

```
/USER/Block/Size 4 2 0.05 m
/USER/Block/Position 0 2 -1.5 m
/USER/Block/Material G4_CONCRETE
/USER/Block/Build

/USER/Block/Size 0.5 2 0.01 m
/USER/Block/Position -4.5 2 -1.5 m
/USER/Block/Material G4_GLASS_PLATE
/USER/Block/Build

/USER/Block/Size 0.05 2 1.5 m
/USER/Block/Position -5 2 0 m
/USER/Block/Material  G4_CONCRETE
/USER/Block/Build

/USER/Block/Size 1 1 0.1 m
/USER/Block/Material G4_Pb
/USER/Block/Position 2 1 2 m
/USER/Block/Build
```

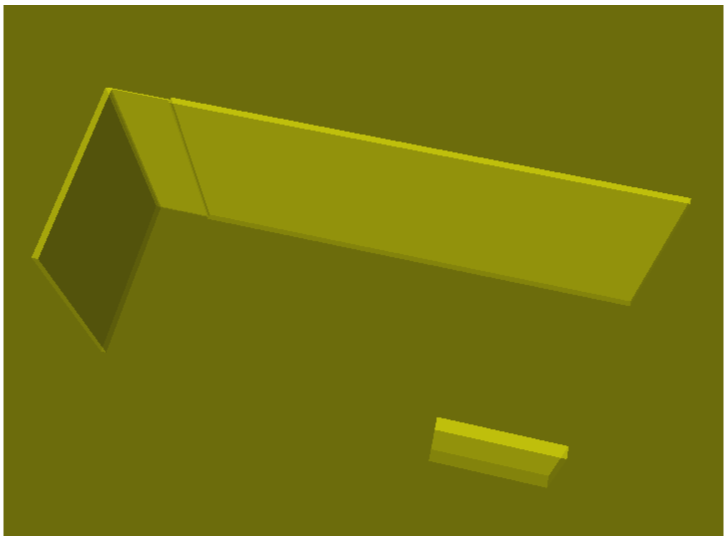

Figure 3: An environment constructed using the USER block commands

These macros that describe the environment can also be read in to MATLAB using the function "**ReadEnvironment.m**". The search algorithm that runs in MATLAB operates in a flat 2D environment, hence the "*y*" dimension will be ignored.

The code for the construction of these block environments is implemented over two files: the Construction is done in "DetectorConstruction.cc", while the interface commands are implemented in "DetectorMessenger.cc".

2.2.3 Increasing Computational Efficiency

GEANT4 tracks and models the interactions of every single particle produced by each generated event, down to a user-definable cut-off point. As a result, a significant amount of computational time and power may be required to perform a run with many events.

In particular for this and future studies, it is intended to use radiation sources that emit in all directions, leading to many events being generated that will not lead to interactions with the detector. One method to improve the efficiency of the simulation is to only track the particles that are likely to hit the detector. To achieve this spatial culling, a "kill zone" has been implemented around the source that only allows particles that are travelling towards the detector – within an acceptance angle – to pass through, while tracking of all other events is discontinued.

The implementation of a kill zone must be carefully defined so as to not eliminate any events that may have eventually generated a count within the detector. This is particularly important when considering the effects of scattering; that is, a generated particle that is initially travelling away from the direction of the detector may be scattered towards, or may generate a secondary particle that is directed towards, the detector by nearby material. To allow for this, all generated particles and secondaries must be tracked over a minimum distance; then scattering of the primary particle or production of a secondary particle with momentum towards the detector may be allowed to continue. This method will still reduce the probability of the detector receiving hits, but in most cases careful tuning of the size of the kill zone and the acceptance angles should minimise losses to be almost unnoticeable whilst still achieving significant improvements in computation time.

The kill zone is defined as a sphere around the source, as shown in Figure 4. The diameter of the sphere may be adjusted by the user. Increasing the size will allow the kill zone to encompass more of the material in the environment surrounding the radiation source, where the majority of the scattering events occur, but will require more computation time. Reducing the size of the kill zone will provide the opposite effect. The angle of acceptance of the kill zone can also be defined by the user, where a larger angle may be necessary where there is material near the detector that may provide significant scattering.

The following user commands allow implementation of a kill zone and fine-tuning of its properties:

/USER/KillZone/set *bool*
where "*bool*" is set to either "true" or "false". This command specifies if the kill zone should be used. The default is true.

/USER/KillZone/Radius *r unit*
where "*r*" is the radius of the kill zone. Set this to be larger if there are scattering objects close to the source. The default is 3m. A unit can also be specified; if left blank, the default unit is meters.

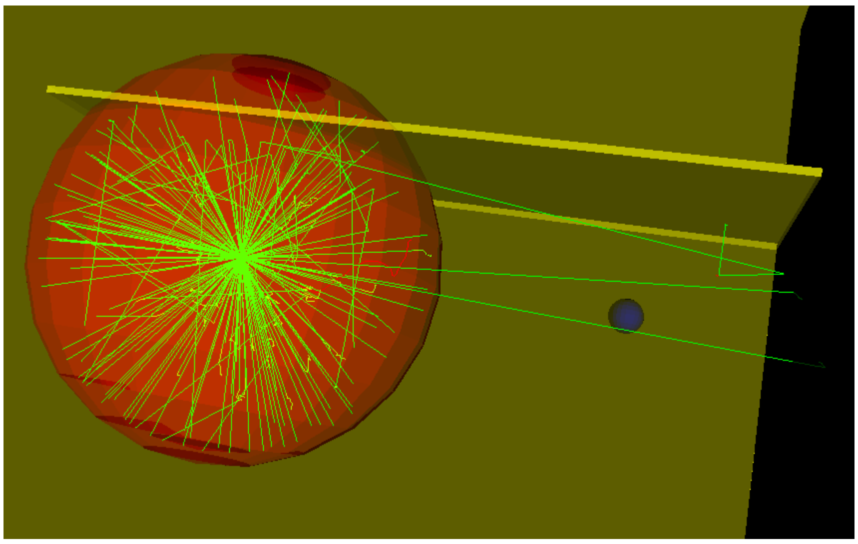

Figure 4: Illustration of the implementation and effects of a "kill zone". The small blue sphere is the detector. The red sphere is a kill zone defined around a radiation source at the boundary of which only particles travelling towards the detector are allowed to pass.

/USER/KillZone/Angle *a*
where "*a*" is an acceptance angle, specifying the allowed deviation around the direction towards the detector. A small angle will reduce the number of particles that leave the kill zone. Use larger angles if there are possible scattering objects close to the detector, of if the detector is obscured by an object.

The code for the kill zone is implemented over three files. The construction of the object is defined in "DetectorConstruction.cc", the interface commands are implemented in "DetectorMessenger.cc" and the decision regarding which particles to let through the kill zone is made in "SteppingAction.cc".

2.2.4 Describing the Source

There are several in-built commands in GEANT4 to describe the particle source; see the "help" command and look under /gps/. Using these commands it is possible to set the position and shape of any source and to define the type and energy of particle to be generated. For example, the following commands set up an isotropic point source of gamma radiation with an energy of 1.3 MeV:

```
/gps/pos/type Point
/gps/particle gamma
/gps/ang/type iso
/gps/energy 1.3 MeV
/gps/pos/centre -3 1 0 m
```

To set up a source distributed within a volume, replace the "Point" command with something similar to:

```
/gps/pos/type Volume
/gps/pos/shape Para
/gps/pos/halfx 0.5 m
/gps/pos/halfy 0.5 m
/gps/pos/halfz 20 m
```

Other shapes such as cylinders and spheres are also available. It is also possible to use a surface as a particle source. Certain volumes within your environment can also be used to describe a source. The distribution of events within this volume as well as the energy levels and energy distribution can also be set in GEANT4. More details on any of these methods can be found in the GEANT4 User Guide or under the "help" command.

More realistic radioactive sources can be modelled by specifying particular radioactive ions and enabling the "G4RadioactiveDecayPhysics ()" physics list which is done within "PhysicsList.cc". This will simulate the radioactive decay processes of the specified isotope and its daughters, generating particles matching the energy distribution and relative abundances expected. For example, to model a caesium-137 source the following commands can be implemented:

```
/gps/particle ion
/gps/ion 55 137 0 0          #Cs137
```

As can be seen, the particle type is changed to ion and then a particular isotope must be specified as the ion; in this case we have specified the /gps/ion to be the caesium-137 isotope. The arguments of the ion command are as follows.

/gps/ion *Z A Q E*
where "*Z*" is the atomic number, "*A*" is the atomic mass, "*Q*" is the ionic charge in units of e, and "*E*" the excitation energy of the ion in keV.

2.2.4.1 Multiple sources

GEANT4 can also simulate multiple sources at one time. However at this stage MATLAB makes separate calls with each source and adds the number of counts at the detector. To add this functionality, multiple kill zones would also need to be implemented. While achievable, implementation will require further development and investigation as to the optimal method for including multiple sources.

2.2.5 Running Events

Once the environment, detector, and radiation sources have been defined, the number of events to run may be specified and the run launched using the following command:

/run/beamOn *n*
where "*n*" is the number of events to run.

When a particle gun is defined, the number of particles generated by the gun will be equivalent to the number of events. On the other hand, when a radioactive ion is specified there may be multiple particles produced per event, as specified by the decay chain of that particular radioisotope.

2.2.6 Converting Text Files from Windows to Linux

Macro files written in Windows and then copied to Linux will sometimes result in an error. This is dependant on which text editors are used and is caused by the Windows text format leaving carriage returns ("\r") at the end of each line. Carriage returns can be removed by using the "tr" command in the following fashion.

```
tr -d '\r' < inputfile > outputfile
```

2.2.7 Visualisation

GEANT4 has many options for visualising the simulation environment and results; information on visualisation can be found in the G4 User Guide. The VRML visualisation option was used during the present work. The VRML driver included in the GEANT4 installation will write its own "**.wrl**" file that can be viewed with any VRML viewer. The "Instant Player" package, which has versions for Windows, Mac and Linux, was found to work well and can be downloaded for free at http://www.instantreality.org/downloads/. From the VRML viewer, the camera position, direction and zoom can all be controlled by the mouse, which is very useful for seeing what GEANT4 is actually doing. Note that when MATLAB runs search algorithms the visualisation is inactive. The visualisation needs to be activated manually. There are several visualisation commands that control the visualisation, which are documented in the GEANT4 User Guide; for an example see Section A.4.2.

2.2.8 Known Bugs

2.2.8.1 Instability over long runs

When performing runs over a large number of events, a problem will sometimes occur where GEANT4 will not close but continues using the CPU. This behaviour occurs inconsistently and the number of events to be run that have been observed to cause this behaviour can be roughly anywhere between 100,000 or 10,000,000 events. This inconsistency and sporadic behaviour has made the problem difficult to debug. A solution to this problem seems to be to run a single event each time before running the full number of events by using the following commands:

```
/run/beamOn 1
/USER/update
/run/beamOn n
```

This solution was originally in place to solve a problem of the kill zone being improperly placed over the source, but it seems to be initialising one or more other components within GEANT4 that is producing more stable behaviour.

2.3 Running a Search Algorithm Test

Using the techniques described in the above sections, everything in this project can be controlled from a MATLAB terminal. In order to run a simulated search to test a search algorithm, the function "**run_mc.m**" is executed. Initialisation data can be edited in "**InitGeantData.m**", being sure to copy the desired environment macro (see section 2.2.2) into the server working directory. To call GEANT4 for a single run with a given source, detector position and environment, "**GetCountsG4.m**" can be executed instead.

3. Simulation Results

3.1 Testing the Search Algorithm

The search algorithm that was used in conjunction with GEANT4 is a particle filter-based algorithm described in [6]. The algorithm uses a sequential Bayesian framework to estimate the sources and carries out observer control by maximising the information gain. The algorithm initialises a large number of "particles" or guesses of the source(s) such that their positions and strengths are distributed according to some suitable prior distributions. Then the likelihood of obtaining the measured data with each of these guessed sources is computed. The particles that are less likely to have given rise to the observed measurements are iteratively eliminated and more likely particles are replicated and jittered to allow better exploration of the solution space. This process is iterated until a desired level of convergence is achieved. The search algorithm considered in this study requires *a priori* specification of the number of sources present and only works with a maximum of two sources. Multiple observers can also be specified; in most of the examples considered here, two observers were used.

The search algorithm is designed to move the observers in a directed way towards and around the sources in order to maximise information gain. Because of the stochastic nature of the algorithm, the observers may take different paths to find the source(s) for the same initial conditions and may even fail to find some sources in some cases (Fig. 5).

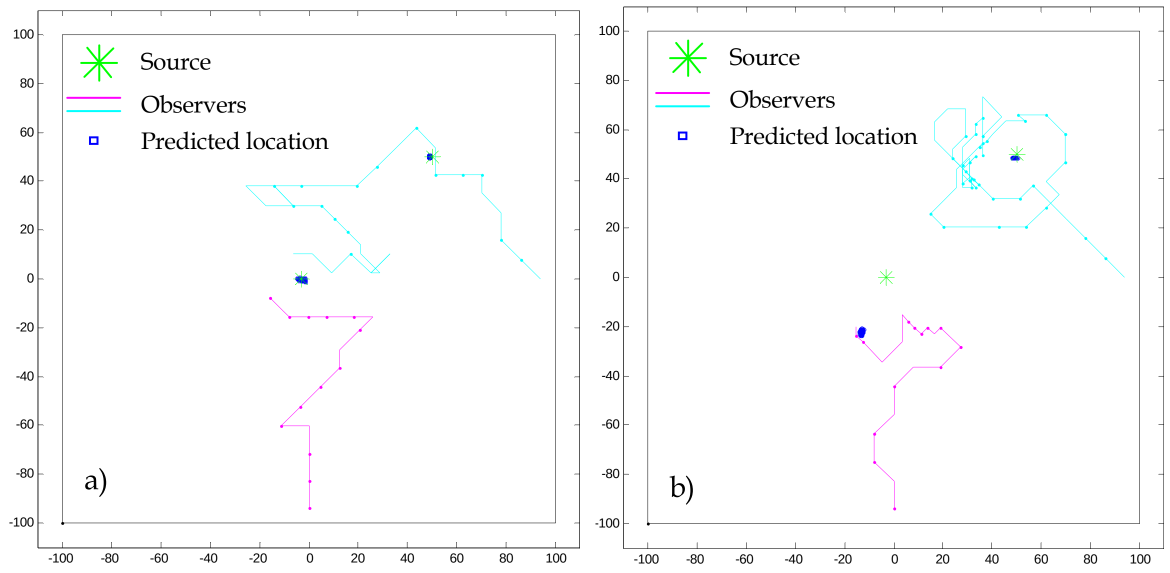

Figure 5: Illustration of the same initial conditions leading to two different search paths in two separate runs. In Figure 5a, both sources are located accurately, while in Figure 5b one source is located incorrectly.

3.2 Effect of Attenuation Models

The application of a search algorithm within a real environment will require the algorithm to handle changes in radiation intensity due to obstructions that attenuate the emitted radiation. It is therefore necessary to include an estimation of attenuation in the estimation of the source strength and position. The MATLAB search algorithm includes, in addition to the application of the simple inverse square law, a simple attenuation factor that will calculate the reduction in the expected counts based on information input regarding the thickness of walls between the source and the detector.

A simulation experiment was performed to compare this simple attenuation model with results achieved in GEANT4. This was done by scanning a line of points beside a source, with a wall positioned to produce attenuation over one half of the line scan (Fig. 6a).

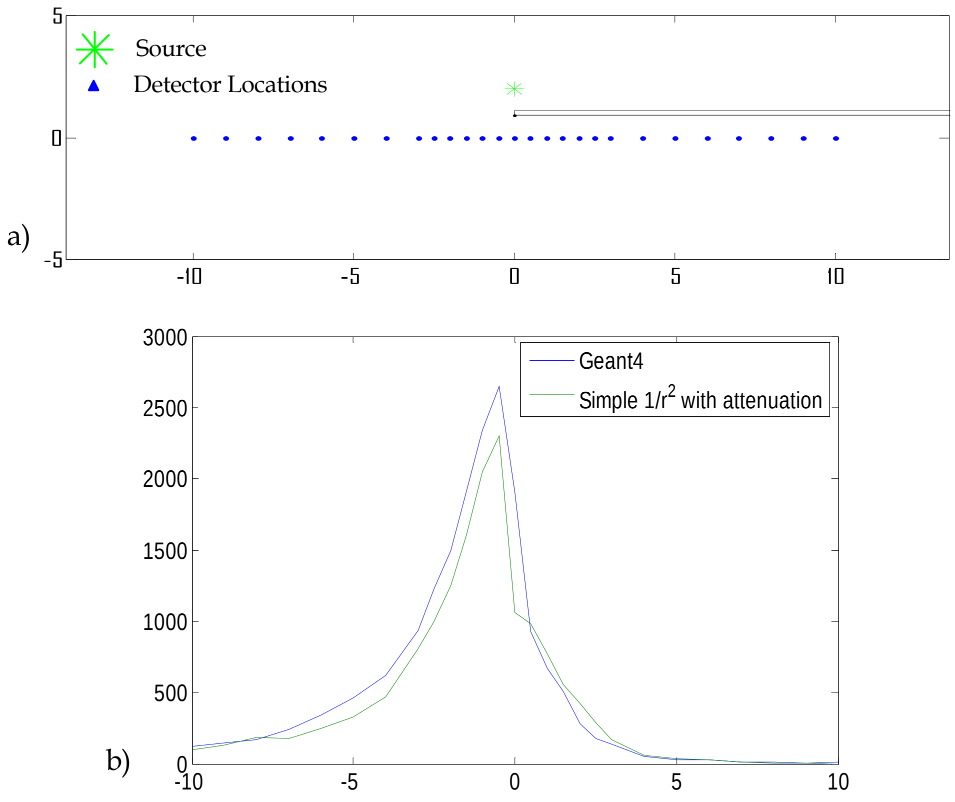

Figure 6: Comparison of the calculation of the effects of an attenuating obstruction on detector counts, as calculated by a GEANT4 simulation and within the MATLAB search algorithm. Figure 6a illustrates the environment construction. Values are in units of metres.

As seen in Figure 6b, attenuation is the dominant source factor that affects counts in such an environment. Other tested environments also showed attenuation to be the main factor. Although this tends to suggest that a simple attenuation based model is sufficient to model the behaviour of a radiation source, there are several important factors lacking in the simplified model that have an effect on the attenuation calculation, such as the ability to easily define a variety of materials and their attenuation factors at different gamma ray energies. GEANT4 will also take into account scattering and the production of secondary particles by a radioactive source such as neutrons and can also model sources other than point sources for which the simplified attenuation and inverse square laws may not be sufficient.

Figure 7 illustrates an example in which an arrangement of blocks of concrete and glass (simulating walls and windows) were constructed near the sources positioned as in the previous search simulation described in Figure 5. The results highlight how the location of the sources has been incorrectly identified as a result of the attenuation effects. While the search algorithm was able to correctly identify the location of both radiation sources in the unobstructed simulation (Fig. 5a), the search algorithm was unable to repeat the success when obstructions were introduced. This result was expected, as the search algorithm has been developed assuming that no obstacles were present [6]. While environments with attenuating obstacles are intractable with the current search algorithms, and significant future effort may be required to develop more advanced search algorithms that are capable of addressing this issue, the present result is nevertheless important because it demonstrated a simulation environment where such new algorithms may be tested as they are developed.

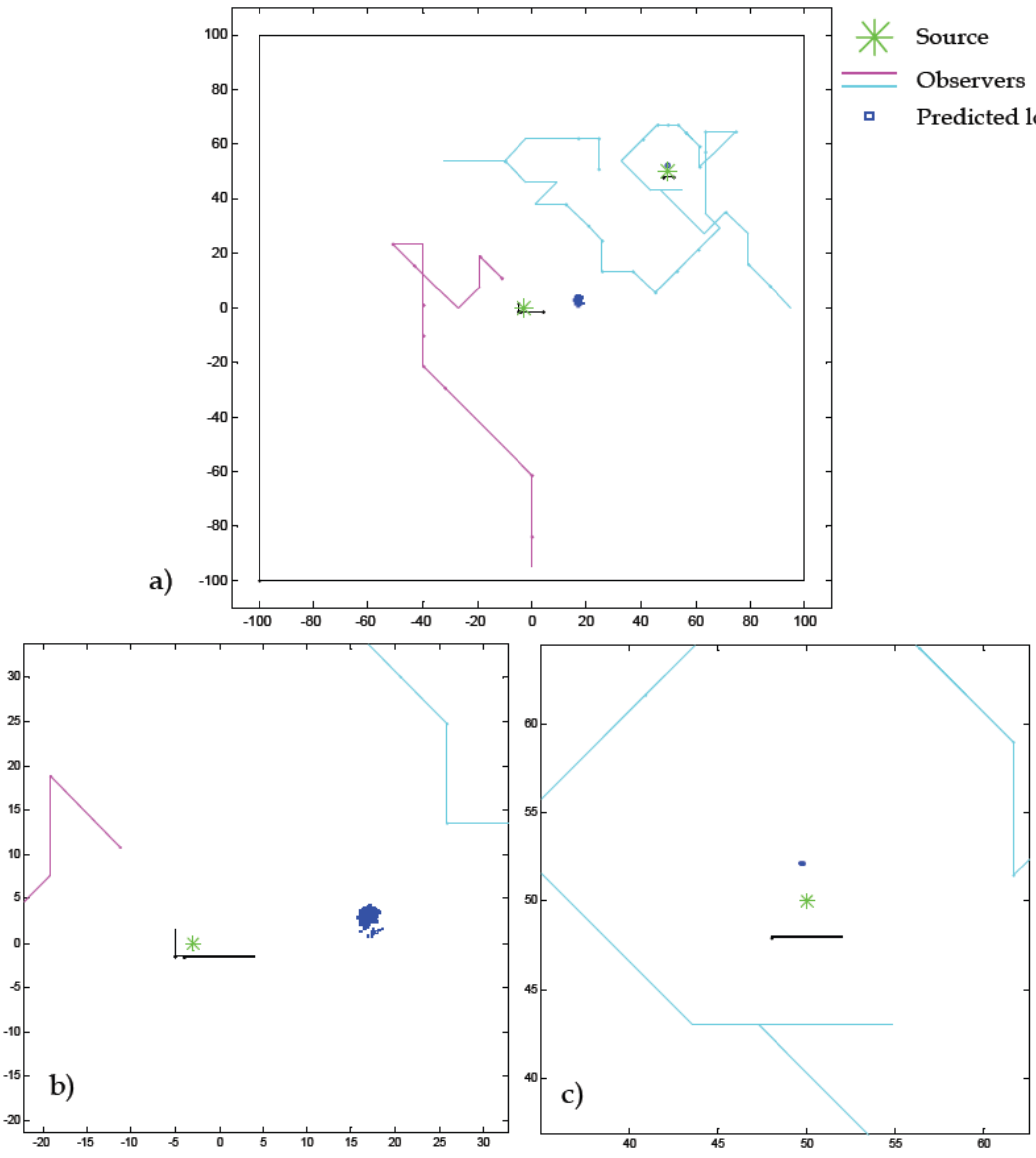

Figure 7: A demonstration of how obstacles may affect the localisation of sources. Figures 7b and 7c are magnified views of Figure 7a, centred on the individual radiation sources.

4. Conclusions

4.1 Capability Summary

By developing an interfacing environment between MATLAB and GEANT4, it is now possible to use MATLAB to rapidly prototype algorithms that require accurate radiation-based modelling. Capabilities have also been added to GEANT4 so that environments can be easily constructed from a command line interface or by other programs through a macro file. The introduction of a tuneable kill zone around the source has also increased the efficiency of the GEANT4 simulations whilst maintaining accuracy in the detector output.

The proof of concept study has demonstrated the ability for a MATLAB-based radiological source search algorithm to interact with a GEANT4-based simulation environment. Because GEANT4 can quite accurately simulate radiation emitted from sources, and also how they interact with the environment, including detectors, this capability reduces the need for field trials to test source localisation algorithms. During the proof of concept study described in this report, the GEANT4 environment was successfully used to test a sequential Bayesian search algorithm. As expected, the algorithm worked well in the absence of attenuating obstacles but failed in their presence. As improved algorithms are developed to search for radiological sources in complex environments with obstacles, source search algorithms may be tested within the simulation environment to rapidly verify their effectiveness.

4.2 Future Development

While this work has taken the first step in establishing a radiological modelling and simulation capability by demonstrating the ability to link GEANT4 and MATLAB to test radiological source search algorithms in realistic complex environments, we recommend future work be undertaken to expand this capability further. One suggested future direction is to improve the useability of the simulation capability by adding a graphical interface to make it easier to add complex 3-D objects to the GEANT environment by sketching the desired shapes without having to specify the objects point by point in a script file.

As demonstrated, there is scope for the use of the simulation capability to assist in further development of more sophisticated search algorithms, in particular to extend them to include distributed as well as point sources.

The feasibility of using GEANT4 for simulating a “real-world” application has been well demonstrated by the proof of concept study. This opens the opportunity for the simulation capability to contribute to further studies. Aside from assisting in further development of more sophisticated search algorithms, the simulation capability can also be applied in a more conventional manner in the modelling of actual or novel theoretical detectors (as opposed to the very generic model currently used), including imaging and directional detectors that could provide alternatives to the use of search algorithms. The simulations could provide useful data to inform a comparative evaluation of these various possible source search modalities, for example by comparing the efficiency/effectiveness of using costly imaging based detectors against using current low-cost in-service equipment in conjunction with search algorithms.

5. References

1. Agostinelli, S., et al. (2003) Geant4 - a simulation toolkit. *Nuclear Instruments and Methods in Physics Research A* **506** 250–303

2. Marinaro, D. G., et al. (2009) *Proceedings of the Joint 6th International Symposium on Protection Against Toxic Substances and 2nd International CBRE Ops Conference* Singapore, 8–11 December 2009

3. Othman, M. A. R., et al. From imaging to dosimetry: GEANT4-based study on the application of Medipix to neutron dosimetry. *Radiation Measurements* **In Press**

4. Gunatilaka, A., Ristic, B. and Morelande, M. (2010) Experimental verification of algorithms for detection and estimation of radioactive sources. In: *13th Int. Conf. Information Fusion,* Edinburgh, UK: 26–29 July, 2010

5. Gunatilaka, A., Ristic, B. and Gailis, R. (2007) *Radiological source localisation*. DSTO-TR-1988

6. Ristic, B., Morelande, M. and Gunatilaka, A. (2010) Information driven search for point sources of gamma radiation. *Signal Processing* **90** (4) 1225–39

7. Ristic, B. and Gunatilaka, A. (2008) Information driven localisation of radiological point sources. *Information Fusion* **9** (2) 317–326

Appendix A: Code

A.1. Code Listings

Most of the computer coding listed below was developed within DSTO. Some programs used open source code available from GEANT4 collaboration and applied relevant modifications. The original open source agreement headers have been left in these modified code listings to acknowledge this fact.

Because the actual source search algorithm code written in MATLAB is not available for public dissemination, only the parts relevant to GEANT4-MATLAB interaction are listed in the appendix.

A.2. MATLAB Code

A.2.1 InitGeantData.m

```
%In this script we initialise variables that control Geant4 and the search
%Algorithm

global GEANT_DATA

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%Data used for Search Algorithm
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
GEANT_DATA.UseGeant = false;

GEANT_DATA.Sources = [-3 0 4000; 50 50 4300]; %[0 2 2000];
GEANT_DATA.EntryPoints = [0 -94; 94 0];

GEANT_DATA.maxStepDistance = 15;
GEANT_DATA.countTime = 2;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%Data used for Server Communications
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
GEANT_DATA.UseVis = false;
%Note: it i required that these macro files are the linux server in the
%WorkDir '/home/linux/g4work'
GEANT_DATA.VisMacro = 'vis_vrml.mac';
GEANT_DATA.EnvironmentMacro = 'BuildEnvironment.mac';

%Server Data
GEANT_DATA.Server.Name = 'geantmg14.dsto.defence.gov.au';
GEANT_DATA.Server.LoginName = 'linux';
GEANT_DATA.Server.WorkDir = '/home/linux/g4work/';
GEANT_DATA.Server.ShellScript = 'run_test_bed.sh';
%Note because we use cygwin commands scp and ssh we refer to C:\Geant4Work
%as the following. Ensure both agree.
GEANT_DATA.LocalWorkDirCyg =  '/cygdrive/c/Geant4Work/'; %local work dir in cygwin format used
for ssh and scp commands
GEANT_DATA.LocalWorkDir = 'C:\Geant4Work\';                %local work dir in win format
GEANT_DATA.OutFile = 'G4Out.csv';

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%Data used for Geant4
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%Hard code the source and detector heights at 1m above the ground
GEANT_DATA.SourceHeight = 1.0;
GEANT_DATA.DetectorHeight = 1.0;
```

```
%Kill zone data
GEANT_DATA.KillZone.set = 'false';
GEANT_DATA.KillZone.radius = '3';
GEANT_DATA.KillZone.angle = '30';

%Source Data
GEANT_DATA.Source.Particle = 'gamma';   %eg gamma neutron ion (ion handles radioactive
sources) many others possible

    %Only used if the particle is described as a radioactive ion
    %First number is atomic number second is atomic mass Describes a Cs 137
    %Source is described below other sources examples are
    %   27 60 0 0   #Co60
    %   92 238 0 0  #U238
    GEANT_DATA.Source.Ion = '55 137 0 0';

GEANT_DATA.Source.Energy = '1.3 MeV';
GEANT_DATA.Source.Direction = 'iso';
GEANT_DATA.Source.type = 'Point';       %Use Volume or Surface to describe a distributed
source

    %This data is only used to describe a distributed Source
    GEANT_DATA.Source.Type.Shape = 'Para';
    % Not all of these parameters are used for every shape.
    %Note: These parameters only describes rectangles spheres and
    %cylinderical volumes and surfaces. Rotation and more complecated
    %shapes are also supported in Geant.
    GEANT_DATA.Source.Type.HalfX = '1 m';
    GEANT_DATA.Source.Type.HalfY = '1 m';
    GEANT_DATA.Source.Type.HalfZ = '1 m';
    GEANT_DATA.Source.Type.Radius = '1 m';
```

A.2.2 GetCountsG4.m

```
function counts = GetCountsG4(SourceXpos, SourceYpos, SourceStrength, DetectorXpos,
DetectorYpos)
% This function runs an instance of Geant 4 to calculate the number
% of counts received by a detector. It should be noted that the source
% strength is the number of particles emitted not the counts from 1m.
% 1,000,000 particles emitted will give 7000-8000 counts to a detector at
% 1m. Currently it takes ~240 (4 min) seconds to run 1,000,000 counts.

    %Initialize the parameters used for Geant4
    InitGeantData;

    sourceHeight = GEANT_DATA.SourceHeight;
    detectorHeight = GEANT_DATA.DetectorHeight;

    G4macro = '';

    %G4macro = '/control/execute C:\\Geant4Work\\BuildEnvironment.mac \n';
    G4macro = [G4macro '/control/execute ' GEANT_DATA.EnvironmentMacro ' \n'];
    if GEANT_DATA.UseVis
        G4macro = [G4macro '/control/execute' GEANT_DATA.VisMacro '\n'];
    end


    G4macro = [G4macro '/gps/pos/type ' GEANT_DATA.Source.type ' \n'];

    if ~strcmp(GEANT_DATA.Source.type, 'Point')
        G4macro = [G4macro '/gps/pos/shape ' GEANT_DATA.Source.Type.Shape ' \n'];
        G4macro = [G4macro '/gps/pos/halfx ' GEANT_DATA.Source.Type.HalfX ' \n'];
        G4macro = [G4macro '/gps/pos/halfy ' GEANT_DATA.Source.Type.HalfY ' \n'];
        G4macro = [G4macro '/gps/pos/halfz ' GEANT_DATA.Source.Type.HalfZ ' \n'];
        G4macro = [G4macro '/gps/pos/radius ' GEANT_DATA.Source.Type.Radius ' \n'];
    end

    G4macro = [G4macro '/gps/particle ' GEANT_DATA.Source.Particle ' \n'];   %Sets particle
type
```

```
    if strcmp(GEANT_DATA.Source.Particle, 'ion')
        G4macro = [G4macro '/gps/ion ' GEANT_DATA.Source.Ion ' \n'];
    else
        G4macro = [G4macro '/gps/energy ' GEANT_DATA.Source.Energy ' \n'];       %Sets the
energy of the radiation
    end

    G4macro = [G4macro '/gps/ang/type ' GEANT_DATA.Source.Direction ' \n'];         %Sets the
distrubition

    G4macro = [G4macro '/gps/pos/centre ' num2str(SourceXpos) ' ' num2str(sourceHeight) ' '
num2str(SourceYpos) ' m \n'];

    G4macro = [G4macro '/USER/DetectorPosition ' num2str(DetectorXpos) ' '
num2str(detectorHeight) ' ' num2str(DetectorYpos) ' m \n'];

    %Sets the kill zone controls
    G4macro = [G4macro '/USER/KillZone/set ' GEANT_DATA.KillZone.set ' \n'];
    G4macro = [G4macro '/USER/KillZone/Radius ' GEANT_DATA.KillZone.radius ' \n'];
    G4macro = [G4macro '/USER/KillZone/Angle ' GEANT_DATA.KillZone.angle ' \n'];

    G4macro = [G4macro '/USER/update \n'];            %Updates the Geometry
    G4macro = [G4macro '/run/beamOn 1 \n' '/USER/update \n'];     % Needed fix instability
problems

    G4macro = [G4macro '/run/beamOn ' num2str(SourceStrength) '\n'];

    G4macroFile = fopen('c:\Geant4Work\matlab_generated.mac','w');
    fprintf(G4macroFile , G4macro);
    fclose(G4macroFile);

    %Commands to the server to copy files and run Geant4
    CopyToServerCom = ['scp ' GEANT_DATA.LocalWorkDirCyg 'matlab_generated.mac '
GEANT_DATA.Server.LoginName '@' GEANT_DATA.Server.Name ':' GEANT_DATA.Server.WorkDir];
    system(CopyToServerCom);
    RunGeantCom = ['ssh ' GEANT_DATA.Server.LoginName '@' GEANT_DATA.Server.Name ' "bash '
GEANT_DATA.Server.WorkDir GEANT_DATA.Server.ShellScript ' matlab_generated.mac"'];
    system(RunGeantCom);
    CopyFromServerCom = ['scp ' GEANT_DATA.Server.LoginName '@' GEANT_DATA.Server.Name ':'
GEANT_DATA.Server.WorkDir GEANT_DATA.OutFile ' ' GEANT_DATA.LocalWorkDirCyg];
    system(CopyFromServerCom);


    G4Output = xlsread([GEANT_DATA.LocalWorkDir GEANT_DATA.OutFile]);

    counts = G4Output(2,5);



end
```

A.2.3 ReadEnvironment.m

```
function Polygons = ReadEnvironment()

    global GEANT_DATA

    %eps = 1e-8;

    Polygons = [];

    Polygons(1,:)= [-100 -100];
    Polygons(2,:)= [-100+eps 100];
    Polygons(3,:)= [100 100+eps];
    Polygons(4,:)= [100+eps -100];
    Polygons(5,:)= [-100 -100];
    Polygons(6,:) = [NaN NaN];
    %system('scp linux@Geantmg14.dsto.defence.gov.au:/home/linux/g4work/BuildEnvironment.mac
/cygdrive/c/Geant4Work/');
    CopyFromServerCom = ['scp ' GEANT_DATA.Server.LoginName '@' GEANT_DATA.Server.Name ':'
GEANT_DATA.Server.WorkDir GEANT_DATA.EnvironmentMacro ' ' GEANT_DATA.LocalWorkDirCyg];
    system(CopyFromServerCom);

    G4Environment = fopen([GEANT_DATA.LocalWorkDir GEANT_DATA.EnvironmentMacro],'r');

    BlockSize = [0 0];
    BlockPosition = [0 0];

    Line = fgetl(G4Environment);
    while ischar(Line)

        if ~isempty(Line)
            LineArray = textscan(Line, '%s %s %s %s %s %s');

            if strcmp(LineArray{1}EA,'/USER/Block/Size');
                BlockSize = [str2double(cell2mat(LineArray{2}))
str2double(cell2mat(LineArray{4}))];

            elseif strcmp(LineArray{1},'/USER/Block/Position');
                BlockPosition = [str2double(cell2mat(LineArray{2}))
str2double(cell2mat(LineArray{4}))];

            elseif strcmp(LineArray{1},'/USER/Block/Build');
                BlockPolygon = AddBlock(BlockSize, BlockPosition);
                Polygons = [Polygons; BlockPolygon];
            end
        end

        Line = fgetl(G4Environment);
    end

    fclose(G4Environment);

end

function BlockPolygon = AddBlock(BlockSize, BlockPosition)

    %eps = 1e-8;

    BlockPolygon(1,:) = [(BlockPosition(1)-BlockSize(1)), (BlockPosition(2)-BlockSize(2))];
    BlockPolygon(2,:) = [(BlockPosition(1)-BlockSize(1)+eps),
(BlockPosition(2)+BlockSize(2))];
    BlockPolygon(3,:) = [(BlockPosition(1)+BlockSize(1)),
(BlockPosition(2)+BlockSize(2)+eps)];
    BlockPolygon(4,:) = [(BlockPosition(1)+BlockSize(1)+eps), (BlockPosition(2)-
BlockSize(2))];
    BlockPolygon(5,:) = [(BlockPosition(1)-BlockSize(1)-eps), (BlockPosition(2)-BlockSize(2)-
eps)];
    BlockPolygon(6,:) = [NaN NaN];

end
```

A.2.4 run_mc.m

```
function [num_found, res, s_time] = run_mc(N,num_obs)
% -------------------------------------------------------------------------

InitGeantData;

spol = ReadEnvironment;


Sources = GEANT_DATA.Sources;
EntryPoints = GEANT_DATA.EntryPoints;

maxStepDistance = GEANT_DATA.maxStepDistance;
```

```
countTime = GEANT_DATA.countTime;

if num_obs == 1
    fid = fopen('res_1.txt','a');
else
    fid = fopen('res_2.txt','a');
end

num_found = 0;
err_x1 = [];
err_x2 = [];
err_y1 = [];
err_y2 = [];
err_I1 = [];
err_I2 = [];
s_time = [];
for n=1:N
    if num_obs == 1
        [found,x_err,y_err,int_err,stime] = pf_search_ultim(spol,[],...
            0,EntryPoints(1,:),maxStepDistance,Sources,1,countTime,2000,3);
    else
        [found,x_err,y_err,int_err,stime] = pf_search_ultim(spol,[],...
            0,EntryPoints,maxStepDistance,Sources,1,countTime,2000,3);
    end
    num_found = num_found + found;
    %fprintf('found %d \n',found);
    if found == 2
        err_x1(n) = x_err(1);
        err_x2(n) = x_err(2);
        err_y1(n) = y_err(1);
        err_y2(n) = y_err(2);
        err_I1(n) = int_err(1);
        err_I2(n) = int_err(2);
        s_time(n) = stime;
    end
    if found == 1
        err_x1(n) = x_err(1);
        err_y1(n) = y_err(1);
        err_I1(n) = int_err(1);
        err_x2(n) = NaN;
        err_y2(n) = NaN;
        err_I2(n) = NaN;
        s_time(n) = stime;
    end
    fprintf('%d    %5.2f   %5.2f   %5.2f   %5.2f   %7.2f   %7.2f    %5.2f\n',...
        found,err_x1(n),err_x2(n),err_y1(n),err_y2(n),err_I1(n),err_I2(n),s_time(n));
    fprintf(fid,'%d    %6.2f   %6.2f   %6.2f   %6.2f   %8.2f   %8.2f    %6.1f\n',...
        found,err_x1(n),err_x2(n),err_y1(n),err_y2(n),err_I1(n),err_I2(n),s_time(n));
end

perc = 100*num_found/(2*N);
RMSE_x(1) = sqrt(mean(err_x1.^2));
RMSE_x(2) = sqrt(mean(err_x2.^2));
RMSE_y(1) = sqrt(mean(err_y1.^2));
RMSE_y(2) = sqrt(mean(err_y2.^2));
RMSE_I(1) = sqrt(mean(err_I1.^2));
RMSE_I(2) = sqrt(mean(err_I2.^2));
res = [err_x1' err_x2' err_y1' err_y2' err_I1' err_I2'];
av_stime = mean(s_time);

fprintf('\n');
fprintf('Number of observers: %d\n',num_obs);
fprintf('Percentage OK: %6.3f \n',perc);
fprintf('RMSE x:  %8.3f   %8.3f \n',RMSE_x(1),RMSE_x(2));
fprintf('RMSE y:  %8.3f   %8.3f \n',RMSE_y(1),RMSE_y(2));
fprintf('RMSE I:  %8.3f   %8.3f \n',RMSE_I(1),RMSE_I(2));
fprintf('avr s.time:  %8.3f  \n',av_stime);
```

A.2.5 get_meas.m

```
function z = get_meas(source,X,Y,mu0,T,gopol,attf)
    % Modified to generate a measurment from multiple sources
    z = zeros(3,1);
    num_sources = size(source,1);

    global GEANT_DATA;

    useGeant = GEANT_DATA.UseGeant;      %true;
    % Distances squared
    if useGeant
        backgroundCounts = poissrnd(mu0*T);
        % A scale factor so we get roughly the same number of counts at 1m
        sourceScaleFactor = 66;
        counts = 0;
        for s=1:num_sources
            tic;
            counts = counts + GetCountsG4(source(s,1), source(s,2),
source(s,3)*sourceScaleFactor*T, X, Y);
            toc;
        end
        z(1) = backgroundCounts + counts;
        z(2) = X;
        z(3) = Y;

    else
        lambda = mu0*T;
        for s=1:num_sources
            Dist_sq(s) = (source(s,1)-X).^2+(source(s,2)-Y).^2;
            obst_dist(s) = obst_thickness(source(s,1:2),[X Y],gopol);
            %fprintf('source: %d    obs-dist: %f\n',s,obst_dist(s));
            lambda = lambda + (exp(-attf*obst_dist(s))*source(s,3)/Dist_sq(s))*T;  %
P(l1)+P(l2) = P(l1+l2)..Poisson
        end
        z(1) =  poissrnd(lambda);
        z(2) = X;
        z(3) = Y;
    end

end


%-----------------------------------------------------------------------
```

A.3. C++ Geant4 Code

A.3.1 test_bed_matlab.cc

```
#include "G4RunManager.hh"
#include "DetectorConstruction.hh"
#include "QGSP_BIC_HP.hh"
#include "PhysicsList.hh"

#include "SteppingAction.hh"

#include "PrimaryGeneratorActionGPS.hh"
#include "G4UIterminal.hh"
#include "G4UItcsh.hh"
//
#include "G4VisExecutive.hh"
//
#include "RunAction.hh"

#include <time.h>
```

```
/*the main function, every C++ program has one*/
int main(int argc, char** argv){


        //choose the Random engine
        CLHEP::HepRandom::setTheEngine(new CLHEP::RanecuEngine());
        //time_t ltime;
        G4int seed=time(0);

        CLHEP::HepRandom::setTheSeed(seed);

        /*
        Geant4: the run manager is the first class that must be instantiated.
        it controls the flow of the program and manages the event loop.
        every simulation has an event loop, iteratively 'shoots' particles, tracks in
geometry, scores energy deposition in some volume, etc
        it also responsible for initialising the simulation, geometry, physics etc but the run
manager must be given this information by the user, see later
        all in-built GEANT4 classes that are used in the simulation are created by the run
manager behind the scenes
        Programming: dynamic memory allocation is used here to create a pointer to a
G4RunManager object and assign memory to the object dynamically (at runtime, see tutorial 3
for more details)
        */
    G4RunManager* rm = new G4RunManager();


    DetectorConstruction* detector = new DetectorConstruction();

    rm->SetUserInitialization(detector);

        // G4VUserPhysicsList* physics = new QGSP_BIC_HP;
        G4VUserPhysicsList* physics = new PhysicsList;

        rm->SetUserInitialization(physics);

        PrimaryGeneratorActionGPS* pga = new PrimaryGeneratorActionGPS();

        rm->SetUserAction(pga);

        //If required, define a RunAction object, pass the pointer to the DetectorConstruction
object to it
        RunAction* ra = new RunAction(detector,pga);
        rm->SetUserAction(ra);

        //If required, define a RunAction object, pass the pointer to the DetectorConstruction
object to it
        SteppingAction* sa = new SteppingAction(ra, detector);

        rm->SetUserAction(sa);

        //Pass a pointer of the particle source to the DetectorConstruction object so it knows
where to build the kill zone
        detector->GiveParticleSource(pga);
        //Initalise the kill zone
        //detector->SetKillZone(true);

        rm->Initialize();

        G4UImanager * UI = G4UImanager::GetUIpointer();

        G4VisManager* visManager = new G4VisExecutive();
        visManager->Initialize();

        //if only 1 command line argeument, we run in interactive mode
    if(argc == 1)
        {
                //AM: vis manager moved into interactive mode only
                //G4VisManager* visManager = new G4VisExecutive();
                //visManager->Initialize();
```

```
		/*this sets up the user interface to run in interactive mode */
		/*comment out for Windows GUI*/
		/*G4UIsession* session = new G4UIterminal(new G4UItcsh);*/
		/* comment out for Linux GUI*/
		G4UIsession* session = new G4UIterminal();
		//UI->ApplyCommand("/control/execute vis.mac");
		session->SessionStart();
		delete session;


    }
	else
	{
		//otherwise we run in batch mode
		G4String command = "/control/execute ";
		G4String fileName = argv[1];
		UI->ApplyCommand(command+fileName);
    }

	delete visManager;
	G4cout << "Visualisation Manager deleted" << G4endl;

	/*the memory that was dynamically allocated for the run manager must be freed*/
	delete rm;
	G4cout << "Run Manager deleted" << G4endl;
    return 0;
}
```

A.3.2 DetectorConstruction.hh

```
//preprocessor directives to prevent multiple includes of the same file
#ifndef DetectorConstruction_H
#define DetectorConstruction_H 1

//preprocessor directive to include the header file for the base class
#include "G4VUserDetectorConstruction.hh"
#include "globals.hh"

#include "G4VPVParameterisation.hh"
#include "G4ThreeVector.hh"
#include "PrimaryGeneratorActionGPS.hh"
#include "G4Colour.hh"

#include <vector>
using namespace std;


//class declaration, we don't need to tell the compiler how big it is, yet
class G4Box;
class G4Orb;
class G4LogicalVolume;
class G4VPhysicalVolume;
class G4Material;
class G4NistManager;
class G4SDManager;
class DetectorMessenger;
class SensitiveDetector;


class G4Para;

struct G4Block {
	G4ThreeVector				Size;
	G4ThreeVector				Position;
	G4Material*					Material;
};
```

```
//this is the user defined class that inherits behaviour from the virtual base class of the
geant4 toolkit
class DetectorConstruction : public G4VUserDetectorConstruction{

public:
	//declare the constructor for this class
    DetectorConstruction();
	//declare the destructor of this class
    ~DetectorConstruction();

	//the one method that MUST be defined in this class, it is called "Construct", takes
no arguments, and returns a pointer to an object of type G4VPhysicalVolume
    G4VPhysicalVolume* ConstructWorld();
	G4VPhysicalVolume* Construct();

	G4NistManager* man;

	G4double GetScoringVolumeMass(){ return ScoringVolumeMass; };
	G4ThreeVector GetPosition(){ return DetectorPos; };

	void UpdateGeometry();
	void ConstructSensitiveDetector(G4LogicalVolume*);
	void ConstructWaterVolume(G4LogicalVolume*);
	void ConstructKillZone(G4LogicalVolume*);
	void SetPosition(G4ThreeVector);
	void ConstructBlock(G4Block, G4LogicalVolume*);
	void AddBlock(G4Block);
	void RemoveAllBlocks();
	const G4ThreeVector& GetPosition() const;
	SensitiveDetector* GetSensitiveDetector() const;
	void GiveParticleSource(PrimaryGeneratorActionGPS*);
	void SetKillZone(bool);
	void SetKillZoneRadius(G4double);
	void SetKillZoneAngle(G4double);
	const G4double GetKillZoneAngle() const;

	G4Box*                         world;           // World
	G4LogicalVolume*       logical_world;
	G4VPhysicalVolume*     physical_world;

	G4Box*                         Box;                   //sub-World
	G4LogicalVolume*       BoxL;
	G4VPhysicalVolume*     BoxP;

	G4double                       killZoneRadius;
	G4double                       killZoneAngle;
	G4bool                         killZone;

private:

	PrimaryGeneratorActionGPS* ParticleSource;

    G4double ScoringVolumeMass;

	G4Box*                         ground;           // Ground
	G4LogicalVolume*       logical_ground;
	G4VPhysicalVolume*     physical_ground;

	G4Sphere*                      KZ;            // Kill Zone for variance reduction
	G4LogicalVolume*       logical_KZ;
	G4VPhysicalVolume*     physical_KZ;

	G4Orb*                         WaterBox1;
	G4LogicalVolume*       logical_WaterBox1;
	G4VPhysicalVolume*     physical_WaterBox1;

	G4Orb*                         scoring;
	G4LogicalVolume*       logical_scoring;
	G4VPhysicalVolume*     physical_scoring;

	G4LogicalVolume*       logical_vol;
```

```
        G4Material* silicon;
        G4Material* air;
        G4Material* water;
        G4Material* concrete;

        G4double XOffset;
        G4double YOffset;
        G4double ZOffset;


        G4double worldX;
        G4double worldY;
        G4double worldZ;

        G4double subWorldX;
        G4double subWorldY;
        G4double subWorldZ;

        G4double water1X;
        G4double water1Y;
        G4double water1Z;

        G4double scoringX;
        G4double scoringY;
        G4double scoringZ;

        G4ThreeVector DetectorPos;

        SensitiveDetector* SD;
        G4SDManager* SDman;

        DetectorMessenger* detectorMessenger;

        vector<G4Block> BlockList;

};
#endif
```

A.3.3 DetectorConstruction.cc

```
//export//preprocessor directive to include header file with class declarations
#include "DetectorConstruction.hh"
//include header files for all classes used in the methods
#include "globals.hh"
#include "G4Element.hh"
#include "G4Material.hh"
#include "G4NistManager.hh"
#include "G4PVPlacement.hh"
#include "G4LogicalVolume.hh"
#include "G4Box.hh"
#include "G4Orb.hh"
#include "G4Sphere.hh"
#include "G4RunManager.hh"
#include "G4RegionStore.hh"
#include "G4UnionSolid.hh"
#include "G4ThreeVector.hh"
#include "G4LorentzRotation.hh"
#include "PrimaryGeneratorActionGPS.hh"

#include "G4GeometryManager.hh"
#include "G4PhysicalVolumeStore.hh"
#include "G4LogicalVolumeStore.hh"
#include "G4SolidStore.hh"
//#include "G4USerLimits.hh"

#include "G4VPVParameterisation.hh"
#include "G4PVDivision.hh"

#include "G4Colour.hh"
```

```
#include "G4VisAttributes.hh"

//for sensitive detector definition
#include "SensitiveDetector.hh"
#include "G4SDManager.hh"

#include "DetectorMessenger.hh"

#include <vector>
using namespace std;

#define PI 3.14159265

//constructor / destructor do nothing
DetectorConstruction::DetectorConstruction()
:world(0),logical_world(0),physical_world(0),
WaterBox1(0),logical_WaterBox1(0),physical_WaterBox1(0),
ground(0),logical_ground(0),physical_ground(0),
KZ(0),logical_KZ(0),physical_KZ(0)


{
// default parameter values here
        worldX = 100. * m;
        worldY = 100. * m;
        worldZ = 100. * m;

        subWorldX = worldX * 0.95;
        subWorldY = worldY * 0.95;
        subWorldZ = worldZ * 0.95;

        G4cout << "World size is " << 2* worldX / cm << " cm" << G4endl;

        // Initial detector position
        XOffset = 2 * m;
        YOffset = 2 * m;
        ZOffset = 0 * m;

        DetectorPos[0] = XOffset;
        DetectorPos[1] = YOffset;
        DetectorPos[2] = ZOffset;

        G4cout << "Position is (" << DetectorPos[0]/m << "," << DetectorPos[1]/m << "," <<
DetectorPos[2]/m << ")" << G4endl;

        water1X = water1Y = water1Z = 25 * cm;     //half-length of volume dimensions

        detectorMessenger = new DetectorMessenger(this);

        killZone = true;
        killZoneRadius = 3*m;
        killZoneAngle = 30;


}

DetectorConstruction::~DetectorConstruction(){
        delete detectorMessenger;
}

/*the virtual method of the Detector Construction class return type is a pointer
 to the G4VPhysicalVolume class it will ultimately be called by the G4RunManager
 class which will define your geometry*/
G4VPhysicalVolume* DetectorConstruction::Construct(){


        /* materials definition */
        //first thing to do is define the materials that will be used in your geometry

        /*define the elements that will be used in our materials*/

        //define hydrogen
        G4double A = 1.01 * g/mole;
        G4double Z = 1;
        G4Element* elH = new G4Element ("Hydrogen", "H", Z, A);

            //define oxygen
        A = 16.0 * g/mole;
        Z = 8;
        G4Element* elO = new G4Element ("Oxygen", "O", Z, A);

            //define nitrogen
        A = 14.01 * g/mole;
        Z = 7;
        G4Element* elN = new G4Element("Nitrogen", "N", Z, A);

            //define carbon
        A = 12.0 * g/mole;
        Z = 6;
        G4Element* elC = new G4Element("Carbon", "C", Z, A);

            //define silicon
        A = 28.09 * g/mole;
        Z = 14;
        G4Element* elSi = new G4Element("Silicon", "Si", Z, A);

            silicon = new G4Material("Silicon", 2.33 * g/cm3, 1);
            silicon ->AddElement(elSi,1);

            /*

            /*define a G4Material object called 'water', assign it the attributes of water*/
            //constructor of the G4Material class requires arguments: string containing name of
    material, density, number of elements
            water = new G4Material("water", 1.0 * g/cm3, 2);

        water->AddElement(elH,2);
        water->AddElement(elO,1);


            /*define air for the world volume*/
        air = new G4Material("dry air", 1.20*mg/cm3, 2);
            //we can also specify the percentage (by mass) composition
        air->AddElement(elN, 75*perCent);
        air->AddElement(elO, 25*perCent);

            /*Use the NIST materials database*/
            man = G4NistManager::Instance();
            man->SetVerbose(1);

            concrete = man->FindOrBuildMaterial("G4_CONCRETE");

            G4Material* elPb = man->FindOrBuildMaterial("G4_Pb");
            G4Material* elU = man->FindOrBuildMaterial("G4_U");
            G4Material* elCs = man->FindOrBuildMaterial("G4_Cs");
            G4Material* elAu = man->FindOrBuildMaterial("G4_Au");

            // G4cout << *(G4Material::GetMaterialTable()) << G4endl;

            /*Construct the world geometries*/

            return ConstructWorld();


    }

    G4VPhysicalVolume* DetectorConstruction::ConstructWorld()
    //void DetectorConstruction::ConstructWorld()
    {

            // Cleanup old geometry
            G4GeometryManager::GetInstance()->OpenGeometry();
            G4PhysicalVolumeStore::GetInstance()->Clean();
            G4LogicalVolumeStore::GetInstance()->Clean();
            G4SolidStore::GetInstance()->Clean();

            G4cout << "Water volume position is (" << DetectorPos[0]/m << "," << DetectorPos[1]/m
    << "," << DetectorPos[2]/m << ")" << G4endl;


            //the whole simulation must be contained within a "world" volume
            //defining a volume requires definition of solid, logical, and physical volume
            //solid volume is the shape, has dimensions
            world = new G4Box("world_box", worldX, worldY, worldZ);

            //logical volume: has a solid volume as a member, a material, last 3???
            logical_world = new G4LogicalVolume(world, air, "world_log", 0,0,0);

            //physical volume: G4PVPlacement class, represents a logical volume that is placed
    somewhere in the mother volume
            physical_world = new G4PVPlacement(0,G4ThreeVector(),logical_world, "world_phys", 0,
    false, 0);

            /* define sub-world volumes here*/

            //sub-World (containment for geometry)
            //define first cut for region here

            //solid volume
            G4Box* Box = new G4Box("Box",subWorldX,subWorldY,subWorldZ);

            //logical volume
        G4LogicalVolume* BoxL = new G4LogicalVolume(Box, air,"BoxL",0, 0, 0);

            //physical volume
        G4VPhysicalVolume* BoxP = new
    G4PVPlacement(0,G4ThreeVector(0,0,0),BoxL,"BoxP",logical_world,false, 0);

            //BoxL->SetVisAttributes(G4VisAttributes::Invisible);


            ground = new G4Box("Ground",subWorldX, subWorldY/2,subWorldZ);

            logical_ground = new G4LogicalVolume(ground, concrete, "ground_log", 0,0,0);

            physical_ground = new G4PVPlacement(0,G4ThreeVector(0,-subWorldY/2,0),logical_ground,
    "ground_phys", BoxL, false, 0);


            /*place a small water volume with an associated Sensitive Detector*/

            ConstructWaterVolume(BoxL);

            //put some visual attributes
            // Define some colours for visulisation
            G4Colour red     (1.0, 0.0, 0.0);
        G4Colour magenta (1.0, 0.0, 1.0);
        G4Colour lblue   (0.0, 0.0, 0.75);
        G4Colour yellow  (1.0, 1.0, 0.0);
        G4Colour green   (0.0, 1.0, 0.0);
        G4Colour cyan    (0.0, 1.0, 1.0);
        G4Colour white   (1.0, 1.0, 1.0);
        G4Colour blue    (0.0, 0.0, 1.0);

            G4VisAttributes* WorldVisAtt = new G4VisAttributes(white);
        WorldVisAtt -> SetVisibility(false);
        //WorldVisAtt -> SetForceWireframe(true);
        logical_world -> SetVisAttributes(WorldVisAtt);

            G4VisAttributes* subWorldVisAtt = new G4VisAttributes(white);
        subWorldVisAtt -> SetVisibility(false);
        //subWorldVisAtt -> SetForceWireframe(true);
        BoxL -> SetVisAttributes(subWorldVisAtt);

            G4VisAttributes* groundVisAtt = new G4VisAttributes(yellow);
        groundVisAtt -> SetVisibility(true);
```

```
	groundVisAtt -> SetForceWireframe(true);
	logical_ground -> SetVisAttributes(groundVisAtt);

	    G4VisAttributes* waterVisAtt = new G4VisAttributes(blue);
	waterVisAtt -> SetVisibility(true);
	waterVisAtt -> SetForceSolid(true);
	logical_WaterBox1 -> SetVisAttributes(waterVisAtt);

	    G4VisAttributes* scoringVisAtt = new G4VisAttributes(white);
	scoringVisAtt -> SetVisibility(false);
	//scoringVisAtt -> SetForceWireframe(true);
	    logical_scoring->SetVisAttributes(scoringVisAtt);

	if (killZone)
	{
		ConstructKillZone(BoxL);
		G4VisAttributes* killzoneVisAtt = new G4VisAttributes(red);
		killzoneVisAtt -> SetVisibility(true);
		killzoneVisAtt -> SetForceWireframe(true);
		logical_KZ -> SetVisAttributes(killzoneVisAtt);
	}
	/*
	G4Block Block;
	Block.Position = G4ThreeVector(10*m, 5*m, 0*m);
	Block.Size = G4ThreeVector(5*m, 5*m, 5*m);
	Block.Material = concrete;
	AddBlock(Block);
	*/
	for (G4int i=0; i < BlockList.size(); i++)
	{
		ConstructBlock(BlockList[i], BoxL);
	}

    return physical_world;

}

void DetectorConstruction::ConstructBlock(G4Block Block, G4LogicalVolume* BoxL)
{
	//G4cout << "Block Size " << Block.Size << "Block Position " << Block.Position <<
"Material " << Block.Material << "Volume " << BoxL << G4endl;

	if (Block.Size[0] == 0 || Block.Size[1] == 0 || Block.Size[2] == 0 )
	{
		G4cout << "Warning block size is 0, block not created" << G4endl;
	} else
	{
		G4Box* block_box = new G4Box("Block", Block.Size[0], Block.Size[1],
Block.Size[2]);

		G4LogicalVolume* logical_block = new G4LogicalVolume(block_box,
Block.Material, "block_log", 0,0,0);

		G4VPhysicalVolume* physical_block = new G4PVPlacement(0, Block.Position,
logical_block, "_phys", BoxL, false, 0);

		G4Colour block_colour  (1.0, 1.0, 0.0);
		G4VisAttributes* blockVisAtt = new G4VisAttributes(block_colour);
		blockVisAtt -> SetVisibility(true);
		blockVisAtt -> SetForceWireframe(true);
		logical_block -> SetVisAttributes(blockVisAtt);
	}

}

void DetectorConstruction::AddBlock(G4Block Block)
{
	BlockList.push_back(Block);
}

void DetectorConstruction::RemoveAllBlocks()
{
        BlockList.clear();
}

void DetectorConstruction::ConstructWaterVolume(G4LogicalVolume* BoxL)
{
        //solid volume
        WaterBox1 = new G4Orb("water_box1", water1X);

        //logical volume
        logical_WaterBox1 = new G4LogicalVolume(WaterBox1, water, "WaterBox1_log", 0,0,0);

        //G4cout << "The water volume will be placed at (" << DetectorPos[0]/m << "," <<
DetectorPos[1]/m << "," << DetectorPos[2]/m << ")" << G4endl;

        //physical volume
        physical_WaterBox1 = new G4PVPlacement(0, DetectorPos, logical_WaterBox1,
"WaterBox1_phys", BoxL, false, 0);

        //logical_WaterBox1->SetVisAttributes(G4VisAttributes(G4Colour(1.0,0.0,1.0)));

        ConstructSensitiveDetector(logical_WaterBox1);
}


void DetectorConstruction::ConstructKillZone(G4LogicalVolume* BoxL)
{

        G4ThreeVector SourcePosition = ParticleSource->GetParticleGun()->GetCurrentSource()-
>GetPosDist()->GetCentreCoords();

        double killZoneThickness = 20*cm;

        G4double kzRmin = killZoneRadius;
        G4double kzRmax = killZoneRadius + killZoneThickness;

        //solid volume
        KZ = new G4Sphere("kill_zone", kzRmin, kzRmax, 0, 2*PI, 0, 2*PI);

        //logical volume
        logical_KZ = new G4LogicalVolume(KZ, air, "KillZone_log", 0,0,0);

        //physical volume
        physical_KZ = new G4PVPlacement(0, SourcePosition, logical_KZ, "KillZone_phys", BoxL,
false, 0);

        /*
        G4Colour killZone_colour (1.0, 0.0, 0.0);
        G4VisAttributes* killzoneVisAtt = new G4VisAttributes(killZone_colour);
    killzoneVisAtt -> SetVisibility(true);
    killzoneVisAtt -> SetForceWireframe(false);
    logical_KZ -> SetVisAttributes(killzoneVisAtt);
        */
}


void DetectorConstruction::UpdateGeometry()
{

  G4RunManager::GetRunManager()->GeometryHasBeenModified();
  G4RunManager::GetRunManager()->DefineWorldVolume(ConstructWorld());
}
void DetectorConstruction::ConstructSensitiveDetector(G4LogicalVolume* logical_vol)
{

        //get pointer to the sensitive detector manager: this class is used by G4RunManager to
see which sensitive detectors there are
    G4SDManager* SDman = G4SDManager::GetSDMpointer();



        G4cout << "Constructing Sensitive Detector" << G4endl;
        scoringX = water1X;
        scoringY = water1Y;
        scoringZ = water1Z;


        /*define a scoring volume within the water volume*/
        scoring = new G4Orb("scoring_box",scoringX);

        //argument list for constructor: solid volume, material, name, ???
        logical_scoring = new G4LogicalVolume(scoring, water, "scoring_log", 0,0,0);

        //assign scoring volume mass
    ScoringVolumeMass = logical_scoring->GetMass();
        G4cout << ScoringVolumeMass / kg << " kg." << G4endl;

        //determine whether the Sensitive Detector already exists
        G4VSensitiveDetector* tgtSD = SDman->FindSensitiveDetector( "SD" );


        //if the sensitive detector is not defined, then Add New Detector. Otherwise, skip and
set appropriate logical volume as SensitiveDetector
        if ( !tgtSD )
        {
                SD = new SensitiveDetector("SD");

                //pass to manager
                SDman->AddNewDetector(SD);
                G4cout << "SD added" << G4endl;
        }

        //now we pass the sensitive detector pointer to the logical volume of our scoring
volume
    logical_scoring->SetSensitiveDetector(SD);

        G4double SDXOffset = 0;
        G4double SDYOffset = 0;
        G4double SDZOffset = 0;


        //argument list for the G4PVPlacement constructor: rotation matrix, position, logical
volume, name, mother logical volumer, boolean, copy number
        physical_scoring = new G4PVPlacement(0, G4ThreeVector(SDXOffset,SDYOffset,SDZOffset),
logical_scoring, "scoring_phys", logical_vol, false, 0);



}

void DetectorConstruction::SetPosition(G4ThreeVector PosVal)
{
        DetectorPos[0] = PosVal[0];
        DetectorPos[1] = PosVal[1];
        DetectorPos[2] = PosVal[2];

        G4double OffsetCheck = DetectorPos[1]/m - water1X/m;

        if(OffsetCheck < 0)
        {
                G4cout << "*** CAUTION: Water volume is below ground level! ***" << G4endl;
                G4cout << "*** Change Y position to be greater than " << water1X/m << " m.
***" << G4endl;
        }

        G4cout << "Changing position to (" << DetectorPos[0]/m << "," << DetectorPos[1]/m <<
"," << DetectorPos[2]/m << ")" << G4endl;

}

inline const G4ThreeVector& DetectorConstruction::GetPosition() const
   { return DetectorPos; }

SensitiveDetector* DetectorConstruction::GetSensitiveDetector() const
```

```
   { return SD; }

// Recieves a pointer for the particle source.
void DetectorConstruction::GiveParticleSource(PrimaryGeneratorActionGPS* gps)
{
        ParticleSource = gps;
}

void DetectorConstruction::SetKillZone(bool kz)
{
        killZone = kz;
}
void DetectorConstruction::SetKillZoneRadius(G4double kzRadius)
{
        killZoneRadius = kzRadius;
}
void DetectorConstruction::SetKillZoneAngle(G4double kzAngle)
{
        killZoneAngle = kzAngle;
}
const G4double DetectorConstruction::GetKillZoneAngle(void) const
{
        return killZoneAngle;
}
```

A.3.4 DetectorMessenger.hh

```
//
// ********************************************************************
// * License and Disclaimer                                           *
// *                                                                  *
// * The  Geant4 software  is  copyright of the Copyright Holders  of *
// * the Geant4 Collaboration. It is provided  under  the terms  and *
// * conditions of the Geant4 Software License,  included in the file *
// * LICENSE and available at  http://cern.ch/geant4/license . These *
// * include a list of copyright holders.                             *
// *                                                                  *
// * Neither the authors of this software system, nor their employing *
// * institutes,nor the agencies providing financial support for this *
// * work  make  any representation or  warranty, express or implied, *
// * regarding  this  software system or assume any liability for its *
// * use. Please see the license in the file  LICENSE  and URL above *
// * for the full disclaimer and the limitation of liability.         *
// *                                                                  *
// * This  code  implementation is the result of  the  scientific and *
// * technical work of the GEANT4 collaboration.                      *
// * By using,  copying,  modifying or  distributing the software (or *
// * any work based  on the software)  you  agree  to acknowledge its *
// * use  in  resulting  scientific  publications,  and indicate your *
// * acceptance of all terms of the Geant4 Software license.          *
// ********************************************************************
//
//
// $Id: ExN07DetectorMessenger.hh,v 1.6 2006/06/29 17:54:44 gunter Exp $
// GEANT4 tag $Name: geant4-09-01-patch-03 $
//

#ifndef DetectorMessenger_h
#define DetectorMessenger_h 1

#include "globals.hh"
#include "G4UImessenger.hh"
#include "G4Material.hh"
#include "G4ThreeVector.hh"

class DetectorConstruction;
class G4UIdirectory;
class G4UIcmdWithAString;
class G4UIcmdWithADouble;
class G4UIcmdWithADoubleAndUnit;
class G4UIcmdWithABool;
class G4UIcmdWithAnInteger;
class G4UIcmdWithoutParameter;
class G4UIcmdWith3Vector;
class G4UIcmdWith3VectorAndUnit;

class G4Material;
class G4NistManager;

class DetectorMessenger: public G4UImessenger
{
  public:
    DetectorMessenger(DetectorConstruction* );
        ~DetectorMessenger();

        G4NistManager* man;

        void SetNewValue(G4UIcommand*, G4String);


    virtual G4String GetCurrentValue(G4UIcommand * command);

  private:

        G4ThreeVector                         BlockSize;
        G4ThreeVector                         BlockPosition;
        G4Material*                                   BlockMaterial;

    DetectorConstruction*             Detector;

    G4UIdirectory*                            Dir;
        G4UIdirectory*                        BlockDir;
        G4UIdirectory*                        KillZoneDir;

        G4UIcmdWithoutParameter*      UpdateCmd;
        G4UIcmdWith3VectorAndUnit*    DetectorPositionCmd;
        G4UIcmdWith3VectorAndUnit*    BlockSizeCmd;
        G4UIcmdWith3VectorAndUnit*    BlockPositionCmd;
        G4UIcmdWithAString*                   BlockMaterialCmd;
        G4UIcmdWithoutParameter*      BlockBuildCmd;
        G4UIcmdWithoutParameter*      BlockRemoveAllCmd;
        G4UIcmdWithABool*                     SetKillZoneCmd;
        G4UIcmdWithADoubleAndUnit*    SetKillZoneRadiusCmd;
        G4UIcmdWithADouble*                   SetKillZoneAngleCmd;

};

#endif
```

A.3.5 DetectorMessenger.cc

```
//
// ********************************************************************
// * License and Disclaimer                                           *
// *                                                                  *
// * The  Geant4 software  is  copyright of the Copyright Holders  of *
// * the Geant4 Collaboration. It is provided  under  the terms  and *
// * conditions of the Geant4 Software License,  included in the file *
// * LICENSE and available at  http://cern.ch/geant4/license . These *
// * include a list of copyright holders.                             *
// *                                                                  *
// * Neither the authors of this software system, nor their employing *
// * institutes,nor the agencies providing financial support for this *
// * work  make  any representation or  warranty, express or implied, *
// * regarding  this  software system or assume any liability for its *
// * use. Please see the license in the file  LICENSE  and URL above *
// * for the full disclaimer and the limitation of liability.         *
// *                                                                  *
// * This  code  implementation is the result of  the  scientific and *
// * technical work of the GEANT4 collaboration.                      *
// * By using,  copying,  modifying or  distributing the software (or *
// * any work based  on the software)  you  agree  to acknowledge its *
// * use  in  resulting  scientific  publications,  and indicate your *
// * acceptance of all terms of the Geant4 Software license.          *
// ********************************************************************
//
//
// $Id: ExN07DetectorMessenger.cc,v 1.6 2006/06/29 17:54:57 gunter Exp $
// GEANT4 tag $Name: geant4-09-01-patch-03 $
//

#include "DetectorMessenger.hh"

#include "DetectorConstruction.hh"
#include "G4UIdirectory.hh"
#include "G4UIcmdWithAString.hh"
#include "G4UIcmdWithABool.hh"
#include "G4UIcmdWithAnInteger.hh"
#include "G4UIcmdWithADouble.hh"
#include "G4UIcmdWith3Vector.hh"
#include "G4UIcmdWithADoubleAndUnit.hh"
#include "G4UIcmdWithoutParameter.hh"
#include "G4UIcmdWith3VectorAndUnit.hh"

#include "G4Material.hh"
#include "G4ThreeVector.hh"

#include "globals.hh"
#include "G4RunManager.hh"
#include "G4NistManager.hh"

DetectorMessenger::DetectorMessenger(DetectorConstruction* Det)
:Detector(Det)
{
        /*Define new Directory to contain user-defined commands*/
  Dir = new G4UIdirectory("/USER/");
  Dir->SetGuidance("UI commands for the dosimeter converter");
  BlockDir = new G4UIdirectory("/USER/Block/");
  BlockDir->SetGuidance("UI commands for building environment components");

  /*Define new commands*/

  // Update command must be called when geometrical values have been changed - re-constructs
the Volumes

  UpdateCmd = new G4UIcmdWithoutParameter("/USER/update",this);
  UpdateCmd->SetGuidance("Update geometry.");
  UpdateCmd->SetGuidance("This command MUST be applied before \"beamOn\" ");
  UpdateCmd->SetGuidance("if you changed geometrical value(s).");
  UpdateCmd->AvailableForStates(G4State_Idle);

  DetectorPositionCmd = new G4UIcmdWith3VectorAndUnit("/USER/DetectorPosition",this);
  DetectorPositionCmd->SetGuidance("Set (x,y,z) coordinates of the water volume in metres.");
  DetectorPositionCmd->SetParameterName("X","Y","Z",false);
  DetectorPositionCmd->AvailableForStates(G4State_Idle);
  DetectorPositionCmd->SetDefaultUnit("m");

  KillZoneDir = new G4UIdirectory("/USER/KillZone/");
  KillZoneDir->SetGuidance("UI commands for setting and tuning the kill zone");

  SetKillZoneCmd = new G4UIcmdWithABool("/USER/KillZone/set", this);
  SetKillZoneCmd->SetGuidance("Set the kill zone flag to ture or false");
  SetKillZoneCmd->AvailableForStates(G4State_Idle);

  SetKillZoneRadiusCmd = new G4UIcmdWithADoubleAndUnit("/USER/KillZone/Radius", this);
  SetKillZoneRadiusCmd->SetGuidance("Set the radius of the kill zone");
  SetKillZoneRadiusCmd->AvailableForStates(G4State_Idle);
  SetKillZoneRadiusCmd->SetDefaultUnit("m");
```

```
    SetKillZoneAngleCmd = new G4UIcmdWithADouble("/USER/KillZone/Angle", this);
    SetKillZoneAngleCmd->SetGuidance("Set the allowed angle of the kill zone in degrees");
    SetKillZoneAngleCmd->AvailableForStates(G4State_Idle);


    man = G4NistManager::Instance();
    man->SetVerbose(1);

    BlockSize = G4ThreeVector(0,0,0);
    BlockPosition = G4ThreeVector(0,0,0);

    BlockMaterial = man->FindOrBuildMaterial("G4_CONCRETE");

    BlockSizeCmd = new G4UIcmdWith3VectorAndUnit("/USER/Block/Size",this);
    BlockSizeCmd->SetGuidance("Set (x,y,z) size of the block.");
    BlockSizeCmd->SetParameterName("X","Y","Z",false);
    BlockSizeCmd->AvailableForStates(G4State_Idle);
    BlockSizeCmd->SetDefaultUnit("m");

    BlockPositionCmd = new G4UIcmdWith3VectorAndUnit("/USER/Block/Position",this);
    BlockPositionCmd->SetGuidance("Set (x,y,z) position of the block center.");
    BlockPositionCmd->SetParameterName("X","Y","Z",false);
    BlockPositionCmd->AvailableForStates(G4State_Idle);
    BlockPositionCmd->SetDefaultUnit("m");

    BlockMaterialCmd = new G4UIcmdWithAString("/USER/Block/Material", this);
    BlockMaterialCmd->SetGuidance("Define block material from G4 materail database. Defult
  G4_CONCRETE");
    BlockMaterialCmd->AvailableForStates(G4State_Idle);

    BlockBuildCmd = new G4UIcmdWithoutParameter("/USER/Block/Build",this);
    BlockBuildCmd->SetGuidance("Builds a block using the current size, position and material");
    BlockBuildCmd->AvailableForStates(G4State_Idle);

    BlockRemoveAllCmd = new G4UIcmdWithoutParameter("/USER/Block/RemoveAll", this);
    BlockRemoveAllCmd->SetGuidance("Removes all blocks from the environment");
    BlockRemoveAllCmd->AvailableForStates(G4State_Idle);

  }

  DetectorMessenger::~DetectorMessenger()
  {
    delete UpdateCmd;

    delete DetectorPositionCmd;

    delete SetKillZoneCmd;
    delete SetKillZoneRadiusCmd;
    delete SetKillZoneAngleCmd;

    delete BlockSizeCmd;
    delete BlockPositionCmd;
    delete BlockBuildCmd;

    delete BlockDir;
    delete KillZoneDir;
    delete Dir;

  }

  void DetectorMessenger::SetNewValue(G4UIcommand* command,G4String newValue)
  {

          /* Define the command action*/
     if (command == UpdateCmd){
             Detector->UpdateGeometry();
     }
     else if (command == DetectorPositionCmd){
             Detector->SetPosition(DetectorPositionCmd->GetNew3VectorValue(newValue));
             Detector->UpdateGeometry();
     }
     else if (command == SetKillZoneCmd) {
```

```
                Detector->UpdateGeometry();
                Detector->SetKillZone(SetKillZoneCmd->GetNewBoolValue(newValue));
                Detector->UpdateGeometry();
        }
        else if (command == SetKillZoneRadiusCmd) {
                Detector->SetKillZoneRadius(SetKillZoneRadiusCmd->GetNewDoubleValue(newValue));
                Detector->UpdateGeometry();
        }
        else if (command == SetKillZoneAngleCmd) {
                Detector->SetKillZoneAngle(SetKillZoneAngleCmd->GetNewDoubleValue(newValue));
        }
        else if (command == BlockSizeCmd){
                BlockSize = (BlockSizeCmd->GetNew3VectorValue(newValue));
        }
        else if (command == BlockPositionCmd){
                BlockPosition = (BlockPositionCmd->GetNew3VectorValue(newValue));
        }
        else if (command == BlockMaterialCmd) {
                  BlockMaterial = man->FindOrBuildMaterial(newValue);
        }
        else if (command == BlockBuildCmd){
                G4Block Block;
                Block.Position = BlockPosition;
                Block.Size = BlockSize;
                Block.Material = BlockMaterial;

                Detector->AddBlock(Block);
                Detector->UpdateGeometry();
        }
        else if (command == BlockRemoveAllCmd) {
                Detector->RemoveAllBlocks();
                Detector->UpdateGeometry();
        }

}

G4String DetectorMessenger::GetCurrentValue(G4UIcommand * command)
{
  G4String ans;

  if(command == DetectorPositionCmd){
          ans = DetectorPositionCmd->ConvertToString(Detector->GetPosition());
  }

  return ans;
}
```

A.3.6 PhysicsList.hh

```
//
// ********************************************************************
// * License and Disclaimer                                           *
// *                                                                  *
// * The  Geant4 software  is  copyright of the Copyright Holders  of *
// * the Geant4 Collaboration. It is provided  under  the terms  and *
// * conditions of the Geant4 Software License,  included in the file *
// * LICENSE and available at  http://cern.ch/geant4/license . These *
// * include a list of copyright holders.                             *
// *                                                                  *
// * Neither the authors of this software system, nor their employing *
// * institutes,nor the agencies providing financial support for this *
// * work  make  any representation or  warranty, express or implied, *
// * regarding  this  software system or assume any liability for its *
// * use. Please see the license in the file  LICENSE  and URL above *
// * for the full disclaimer and the limitation of liability.         *
// *                                                                  *
// * This  code  implementation is the result of  the  scientific and *
// * technical work of the GEANT4 collaboration.                      *
// * By using,  copying,  modifying or  distributing the software (or *
// * any work based  on the software)  you  agree  to acknowledge its *
// * use  in  resulting  scientific  publications,  and indicate your *
// * acceptance of all terms of the Geant4 Software license.          *
// ********************************************************************
//
//  Based on exrdmPhysicsList.hh
//....oooOO0OOooo........oooOO0OOooo........oooOO0OOooo........oooOO0OOooo......

#ifndef PhysicsList_h
#define PhysicsList_h 1

#include "G4VModularPhysicsList.hh"
#include "globals.hh"
#include <vector>


class G4VPhysicsConstructor;


//....oooOO0OOooo........oooOO0OOooo........oooOO0OOooo........oooOO0OOooo......

class PhysicsList: public G4VModularPhysicsList
{
public:
  PhysicsList();
  virtual ~PhysicsList();

  void ConstructParticle();

  void SetCuts();
  void SetCutForGamma(G4double);
  void SetCutForElectron(G4double);
  void SetCutForPositron(G4double);

  void ConstructProcess();

private:

  void AddExtraBuilders(G4bool flagHP);

  // hide assignment operator
  PhysicsList & operator=(const PhysicsList &right);
  PhysicsList(const PhysicsList&);

  G4double cutForGamma;
  G4double cutForElectron;
  G4double cutForPositron;

  G4VPhysicsConstructor*  emPhysicsList;
  G4VPhysicsConstructor*  raddecayList;
  G4VPhysicsConstructor*  particleList;
  G4VPhysicsConstructor*  hadPhysicsList;


  std::vector<G4VPhysicsConstructor*>  hadronPhys;
  G4int nhadcomp;


};

//....oooOO0OOooo........oooOO0OOooo........oooOO0OOooo........oooOO0OOooo......

#endif
```

A.3.7 PhysicsList.cc

```
//
// ********************************************************************
// * License and Disclaimer                                           *
// *                                                                  *
// * The  Geant4 software  is  copyright of the Copyright Holders  of *
// * the Geant4 Collaboration. It is provided  under  the terms  and *
// * conditions of the Geant4 Software License,  included in the file *
// * LICENSE and available at  http://cern.ch/geant4/license . These *
// * include a list of copyright holders.                             *
// *                                                                  *
// * Neither the authors of this software system, nor their employing *
// * institutes,nor the agencies providing financial support for this *
// * work  make  any representation or  warranty, express or implied, *
// * regarding  this  software system or assume any liability for its *
// * use. Please see the license in the file  LICENSE  and URL above *
// * for the full disclaimer and the limitation of liability.         *
// *                                                                  *
// * This  code  implementation is the result of  the  scientific and *
// * technical work of the GEANT4 collaboration.                      *
// * By using,  copying,  modifying or  distributing the software (or *
// * any work based  on the software)  you  agree  to acknowledge its *
// * use  in  resulting  scientific  publications,  and indicate your *
// * acceptance of all terms of the Geant4 Software license.          *
// ********************************************************************
//
//      Based on exrdmPhysicsList.cc
//....oooOO0OOooo........oooOO0OOooo........oooOO0OOooo........oooOO0OOooo......

#include "PhysicsList.hh"

#include "G4EmStandardPhysics.hh"
#include "G4RegionStore.hh"
#include "G4Region.hh"
#include "G4ProductionCuts.hh"
#include "G4ProcessManager.hh"
#include "G4ParticleTypes.hh"
#include "G4ParticleTable.hh"

#include "G4Gamma.hh"
#include "G4Electron.hh"
#include "G4Positron.hh"

#include "G4UnitsTable.hh"
#include "G4LossTableManager.hh"


#include "HadronPhysicsQGSP_BIC.hh"
//#include "HadronPhysicsQGSP_BIC_HP.hh"

#include "G4EmExtraPhysics.hh"
#include "G4HadronElasticPhysics.hh"
#include "G4QStoppingPhysics.hh"
#include "G4IonBinaryCascadePhysics.hh"
#include "G4RadioactiveDecayPhysics.hh"
#include "G4NeutronTrackingCut.hh"
#include "G4DecayPhysics.hh"

//....oooOO0OOooo........oooOO0OOooo........oooOO0OOooo........oooOO0OOooo......

PhysicsList::PhysicsList() : G4VModularPhysicsList()
{
  G4LossTableManager::Instance();
  defaultCutValue = 1.*mm;
  cutForGamma     = defaultCutValue;
  cutForElectron  = defaultCutValue;
  cutForPositron  = defaultCutValue;
```

```
    SetVerboseLevel(0);


    //default physics
    particleList = new G4DecayPhysics();

    G4cout << "Particle List done" << G4endl;

    //default physics
    raddecayList = new G4RadioactiveDecayPhysics();

    G4cout << "Rad Decay List done" << G4endl;

    // EM physics
    emPhysicsList = new G4EmStandardPhysics();

          G4cout << "Em Physics List done" << G4endl;

    // Had physics
      AddExtraBuilders(false);
  //      hadPhysicsList = new HadronPhysicsQGSP_BIC_HP("std-hadron");
          hadPhysicsList = new HadronPhysicsQGSP_BIC("std-hadron");

  //  AddExtraBuilders(true);
  //  hadPhysicsList = new exrdmPhysListHadron("hadron");
  //  nhadcomp = 0;

    G4cout << "Hadron Physics List done " << nhadcomp << G4endl;
  }

  //....oooOO0OOooo........oooOO0OOooo........oooOO0OOooo........oooOO0OOooo......

  PhysicsList::~PhysicsList()
  {
   // delete pMessenger;
    delete raddecayList;
    delete emPhysicsList;
    if (hadPhysicsList) delete hadPhysicsList;

    //AM removed nhadcomp and replaced with hadronPhys.size()
    if (hadronPhys.size() > 0)
    {
      for(G4int i = 0; i < hadronPhys.size(); i++)
          {
            delete hadronPhys[i];
          }
    }

    /*if (nhadcomp > 0) {
      for(G4int i=0; i<nhadcomp; i++) {
        delete hadronPhys[i];
      }
    }*/

  }

  //....oooOO0OOooo........oooOO0OOooo........oooOO0OOooo........oooOO0OOooo......

  void PhysicsList::ConstructParticle()
  {
    particleList->ConstructParticle();
  }

  //....oooOO0OOooo........oooOO0OOooo........oooOO0OOooo........oooOO0OOooo......

  void PhysicsList::ConstructProcess()
  {
    AddTransportation();
    // em
    emPhysicsList->ConstructProcess();
    // decays
    particleList->ConstructProcess();
    raddecayList->ConstructProcess();
    // had
    if (hadronPhys.size() > 0) {
      for(G4int i=0; i<hadronPhys.size(); i++) {
        (hadronPhys[i])->ConstructProcess();
      }
    }
    if (hadPhysicsList) hadPhysicsList->ConstructProcess();


  }


  void PhysicsList::AddExtraBuilders(G4bool flagHP)
  {


    if (emPhysicsList) delete emPhysicsList;
    emPhysicsList = new G4EmStandardPhysics();
    G4cout << "New EM Phys List - G4EmStandardPhysics" << G4endl;

    /*
    if (hadPhysicsList) {
      delete hadPhysicsList;
      hadPhysicsList = 0;
    }*/
    //nhadcomp = 6;

    //verboseLevel = 2;

    hadronPhys.push_back( new G4EmExtraPhysics("extra EM"));
    hadronPhys.push_back( new G4HadronElasticPhysics("elastic",verboseLevel,flagHP));
    hadronPhys.push_back( new G4QStoppingPhysics("stopping",verboseLevel));
    hadronPhys.push_back( new G4IonBinaryCascadePhysics("ionBIC"));
    //hadronPhys.push_back( new G4RadioactiveDecayPhysics("radioactiveDecay"));
    hadronPhys.push_back( new G4NeutronTrackingCut("Neutron tracking cut"));
  }

  //....oooOO0OOooo........oooOO0OOooo........oooOO0OOooo........oooOO0OOooo......

  void PhysicsList::SetCuts()
  {

    SetCutValue(cutForGamma, "gamma");
    SetCutValue(cutForElectron, "e-");
    SetCutValue(cutForPositron, "e+");
    G4cout << "world cuts are set" << G4endl;

    if (verboseLevel>0) DumpCutValuesTable();
  }

  //....oooOO0OOooo........oooOO0OOooo........oooOO0OOooo........oooOO0OOooo......

  void PhysicsList::SetCutForGamma(G4double cut)
  {
    cutForGamma = cut;
  }

  //....oooOO0OOooo........oooOO0OOooo........oooOO0OOooo........oooOO0OOooo......

  void PhysicsList::SetCutForElectron(G4double cut)
  {
    cutForElectron = cut;
  }

  //....oooOO0OOooo........oooOO0OOooo........oooOO0OOooo........oooOO0OOooo......

  void PhysicsList::SetCutForPositron(G4double cut)
  {
    cutForPositron = cut;
  }
```

A.3.8 PrimaryGeneratorActionGPS.hh

```
#ifndef PrimaryGeneratorActionGPS_hh
#define PrimaryGeneratorActionGPS_hh 1

//as this user defined class is derived from a geant4 base class, we need to include the class
definition
#include "G4VUserPrimaryGeneratorAction.hh"
#include "G4ParticleGun.hh"
#include "G4GeneralParticleSource.hh"

//this class is used to specify details of each particle, referred to as a 'generator'
class G4ParticleGun;
class G4GeneralParticleSource;
class G4Event;//geant4 class which represents an event, ie. particle history

//the primary generator action class is used to specify how each primary particle is generated

class PrimaryGeneratorActionGPS : public G4VUserPrimaryGeneratorAction {

public:
    PrimaryGeneratorActionGPS();
    ~PrimaryGeneratorActionGPS();

//you must define this method, it is called by the G4RunManager
//run manager passes the pointer to an event object, it will be given attributes from the
Particle Gun
    void GeneratePrimaries(G4Event*);
// virtual G4ParticleGun* GetParticleGun() {return gun;};
virtual G4GeneralParticleSource* GetParticleGun() {return gun;};

private:
//private member of this class, a pointer to an object of another class
    //G4ParticleGun* gun;

        G4GeneralParticleSource* gun;

};

#endif
```

A.3.9 PrimaryGeneratorActionGPS.cc

```
#include "PrimaryGeneratorActionGPS.hh"

//include class definition for the particle gun
#include "G4GeneralParticleSource.hh"


#include "G4Event.hh"


#include "Randomize.hh"

PrimaryGeneratorActionGPS::PrimaryGeneratorActionGPS(){

//use dynamic memory allocation for the G4GeneralParticleSource object


        gun = new G4GeneralParticleSource();

}

PrimaryGeneratorActionGPS::~PrimaryGeneratorActionGPS(){
//free the dynamically allocated memory
    delete gun;

}
```

```
#include "G4Event.hh"
//this method will be called by the RunManager at the beginning of each particle history
void PrimaryGeneratorActionGPS::GeneratePrimaries(G4Event* anEvent){


    gun->GeneratePrimaryVertex(anEvent);

}
```

A.3.10 RunAction.hh

```
#ifndef RunAction_hh
#define RunAction_hh 1
//
#include "G4UserRunAction.hh"
#include "G4UnitsTable.hh"
#include "DetectorConstruction.hh"
#include "G4ParticleGun.hh"
#include "PrimaryGeneratorActionGPS.hh"
/*#ifdef G4ANALYSIS_USE_ROOT
#include "c:/root/include/TFile.h"
#include "c:/root/include/TH3D.h"
#endif
*/
#include <fstream>
#include <sys/stat.h>

//declare the DetectorConstruction class as we will define a pointer later
class DetectorConstruction;
class PrimaryGeneratorActionGPS;

//needed for using standard libraries
using namespace std;

//run action class, carries out tasks at the begin and end of each run
//the concept of a run incorporates a fixed geometry, fixed beam conditions, simulation of
number of primaries
//begins with /run/beamOn command and finishes with tracking of last secondary to zero energy

class RunAction : public G4UserRunAction {
//
public:
//run action class needs pointer ot the detector construction class in order to get details of
the readout geometry
//accepts pointer to detector construction class

    RunAction(DetectorConstruction*,PrimaryGeneratorActionGPS*);
    ~RunAction();


    //
public:
//note argument of these methods is a pointer to a G4Run object
    void BeginOfRunAction(const G4Run*);
    void EndOfRunAction(const G4Run*);

//this is called by the event action class and passes an array of G4double variables. Gives
dose in each grid box per primary particle
    void ProcessDoseEvent(const G4double);
        void ProcessStep(const G4double);


private:
//an ofstream to access the output file
    ofstream  outfile;

//      G4String rootFilename;
        G4String inum;
        G4String jnum;
        G4String FilenameBase;
  //    G4String date;
/*      G4String HistoName;
        G4String HistoFilenameBase;
        G4String rootFilenameExtension;
        G4String histoNum;
*/
        G4int N;

/*
        TFile* fFile;
        TH1D* fpHisto;
        TH1D* fEHisto;
*/

//dose in each grid boxe
    G4double  Dose;

//the square of the dose deposited in each grid box, needed for error calculation
    G4double  DoseSquared;

        // The number of counts through a dector
        G4double Counts;

//local pointer for detector construction class
    DetectorConstruction* Detector;
        DetectorConstruction* myDetector;
        PrimaryGeneratorActionGPS* myPGA;
        PrimaryGeneratorActionGPS* pga;
        //G4ParticleGun* particleDef;

        G4double NumberOfEvents;
//      G4double Fluence;
        G4double energy;

        G4ThreeVector Posn;
        G4double PosX;
        G4double PosY;
        G4double PosZ;


};

#endif
```

A.3.11 RunAction.cc

```
//
#include "RunAction.hh"
#include "G4Run.hh"
//
#include "DetectorConstruction.hh"
#include "G4ParticleGun.hh"
#include "PrimaryGeneratorActionGPS.hh"

#include <stdio.h>
//#include <time.h>


RunAction::RunAction(DetectorConstruction* Detector,PrimaryGeneratorActionGPS* pga){
//
    Dose = 0;
    DoseSquared = 0;
        Counts = 0;

        //take the DetectorConstruction pointer given when this object is created (in main)
and copy to local member
    myDetector = Detector;
        myPGA = pga;
        //Initialise the Histogram alphanumeric ID
        N = 0+96; //lowercase alpha
        /*
        histoNum = N;
        */

        G4int i = 0+48; //ASCII value of '0' is 48, so 'i' is i + 48, i < 10
        G4int j = 0+48;
        inum = i;
        jnum = j;
        G4String Directory = "";//"c:\\Geant4work\\";
        FilenameBase = "G4Out";
        G4String FilenameExtension = ".csv";
        //G4String Filename = Directory + FilenameBase + "_" + inum + jnum +
FilenameExtension;
        G4String Filename = Directory + FilenameBase + FilenameExtension; // Overwrites old
output files.

        //check whether Filename already exists
        struct stat stFileInfo;
        G4bool blnReturn;
        G4int intStat;

         // Attempt to get the file attributes
        intStat = stat(Filename.c_str(),&stFileInfo);
        if(intStat == 0) {
                // We were able to get the file attributes so the file obviously exists.
                blnReturn = true;
        } else {
                // We were not able to get the file attributes. This may mean that we don't
have permission to
                // access the folder which contains this file. If you need to do that level of
checking, lookup the
                // return values of stat which will give you more details on why stat failed.
                blnReturn = false;
        }

        if(!blnReturn){
                outfile.open(Filename,ios::out|ios::app);
                G4cout << "File did NOT exist, file " << Filename << " created." << G4endl;

        }
        else
        {
                outfile.open(Filename,ios::out);
                G4cout << "File EXISTED, overwrite " << Filename << G4endl;
        }
        /*
        else while (blnReturn){
                if(j==57){
                        j = 0+48;
                        i++;
                        inum = i;
                }
                else{j++;}

                jnum = j;


                Filename = Directory + FilenameBase + "_" + inum + jnum + FilenameExtension;;

                //Filename = Directory + FilenameBase + FilenameExtension;; // Overwrites old
output files.
                intStat = stat(Filename.c_str(),&stFileInfo);

                if(intStat == 0) {
                        // We were able to get the file attributes so the file obviously
exists.
                        blnReturn = true;
                        }
                else {
                        // We were not able to get the file attributes. This may mean that we
don't have permission to
                        // access the folder which contains this file. If you need to do that
level of checking, lookup the
                        // return values of stat which will give you more details on why stat
failed.

                        outfile.open(Filename,ios::out|ios::app);
                        G4cout << "File EXISTED, new file " << Filename << " created." <<
G4endl;
                        blnReturn = false;
                }
        }
        */
        outfile << "Energy (MeV), X (m), Y (m), Z (m), Counts, TotalAbsorbed Dose (J/kg),
Absorbed Dose per Event (J/kg), Dose Error per Event(J/kg), Events" << G4endl;
        //outfile << "Energy (MeV), X (m), Y (m), Z (m), TotalAbsorbed Dose (J/kg), Absorbed
Dose per Event (J/kg), Dose Error per Event(J/kg), Events" << G4endl;

        /*#ifdef G4ANALYSIS_USE_ROOT

        //Set up files for use of ROOT analysis package

        HistoFilenameBase = FilenameBase;
        rootFilenameExtension = ".root";
        rootFilename = HistoFilenameBase + "_" + inum + jnum + rootFilenameExtension;
        G4cout << "Root file name " << rootFilename << " created." << G4endl;



        // Attempt to get the file attributes
        intStat = stat(rootFilename.c_str(),&stFileInfo);
        if(intStat == 0) {
                // We were able to get the file attributes so the file obviously exists.
                blnReturn = true;
                } else {
                        // We were not able to get the file attributes. This may mean that we
don't have permission to
                        // access the folder which contains this file. If you need to do that
level of checking, lookup the
                        // return values of stat which will give you more details on why stat
failed.
                        blnReturn = false;
                }
                if(blnReturn) {
                        G4cout << "**Warning, Histo file " << rootFilename << " already
existed**" << G4endl;
                }

                fFile = new TFile(rootFilename,"CREATE");
        #endif
        */
}

RunAction::~RunAction(){

        //close the output file when this class is destroyed
        outfile.close();
}

void RunAction::BeginOfRunAction(const G4Run* aRun){

        //initialise the dose and dose squared for this run
    Dose = 0;
    DoseSquared = 0;
        Counts = 0;


        /*
        #ifdef G4ANALYSIS_USE_ROOT
          //Only required if ROOT is being used
                N++;
                histoNum = N;
                HistoName = "Dose" + histoNum;
                fpHisto = new TH1D(HistoName,"Dose deposition in water",50,0,3000);

                HistoName = "Energy" + histoNum;
                fEHisto = new TH1D(HistoName,"Energy of particle entering water
volume",50,0,3000);
                G4cout << "Histograms initialised - " << HistoName << G4endl;
        #endif
        */
}

void RunAction::ProcessStep(G4double En){
        /*#
        ifdef G4ANALYSIS_USE_ROOT
                if(En > 0 *keV)
                {
                        fEHisto->Fill(En/keV);
                }
        #endif
        */
}

void RunAction::ProcessDoseEvent(const G4double DosePerEvent){
//here we receive the dose deposited for a single event, add to accumulative depth dose and
the accumulated dose per primary squared


        if(DosePerEvent > 0 *keV)
        {
                //increment the dose deposited in this  box
                Dose += DosePerEvent;
                //increment the sum of the dose in this box squared
                DoseSquared += pow(DosePerEvent,2);

                Counts++;

                /*
                #ifdef G4ANALYSIS_USE_ROOT
                fpHisto->Fill(DosePerEvent/keV);
                #endif
                */
        }


}

//task to be carried out at the end of the run
void RunAction::EndOfRunAction(const G4Run* aRun){
//

    //get the number of primary particles being simulated for this run
    NumberOfEvents = aRun->GetNumberOfEventToBeProcessed();

    //first, calculation average energy deposition per event along with errors


        //average energy deposition
    G4double AverageEnergy = Dose / NumberOfEvents;
        //calculate the average dose squared in this grid box
    G4double AverageEnergySquared = DoseSquared / NumberOfEvents;

        //calculate the square of the average dose
    G4double SquaredAverageEnergy = pow(AverageEnergy, 2);
        //calculate the standard deviation in the average dose squared
    G4double StandardDeviation = sqrt(AverageEnergySquared - SquaredAverageEnergy);
        //calculate the standard error in the average dose estimate, 95% confidence limits
    G4double EnergyError = 2 * StandardDeviation / sqrt(G4double(NumberOfEvents));

        //second, calculate the average Dose (in joules) per event along with error estimate
        //convert energy deposition from MeV to joules
    G4double EnergyInJoules = AverageEnergy / joule;
        G4double TotalEnergyInJoules = Dose / joule;

        //get the mass of the scoring volume
    G4double MassInKilos = myDetector->GetScoringVolumeMass()/kg;

        //      G4cout << "MassInKilos " << MassInKilos / kg << G4endl;
    //
    G4double AbsorbedDose = EnergyInJoules / MassInKilos; // joule per kilo
        G4double TotalAbsorbedDose = TotalEnergyInJoules / MassInKilos; // joule per kilo

        //calculate Dose error from energy deposition error
    G4double DoseError = EnergyError / joule / MassInKilos;


        //average number of events
    //G4double AverageCount = Counts / NumberOfEvents;

        energy = myPGA->GetParticleGun()->GetParticleEnergy();

        Posn = myDetector->GetPosition();
        PosX = Posn[0];
        PosY = Posn[1];
        PosZ = Posn[2];


        //print message to screen
    G4cout << "***********************************************" << G4endl;
    G4cout << "*  Total Dose (Gy)    Error (Gy)                  *" << G4endl;
    G4cout << "***********************************************" << G4endl;
        //print the result to screen
    G4cout << TotalAbsorbedDose / (joule/kg) << " " << DoseError / (joule/kg) << G4endl;



    if(outfile)
                outfile << energy << "," << PosX/m << "," << PosY/m << "," << PosZ/m << "," <<
Counts << "," << TotalAbsorbedDose/ (joule/kg) << "," << AbsorbedDose/ (joule/kg) << "," <<
DoseError/ (joule/kg) << "," << NumberOfEvents << G4endl;
        //if(outfile)
        //      outfile << energy << "," << PosX/m << "," << PosY/m << "," << PosZ/m << "," <<
TotalAbsorbedDose/ (joule/kg) << "," << AbsorbedDose/ (joule/kg) << "," << DoseError/
(joule/kg) << "," << NumberOfEvents << G4endl;

        /*
        #ifdef G4ANALYSIS_USE_ROOT
                fFile->Write();
        #endif
        */

}
```

A.3.12 SensitiveDetector.hh

```
/*G4VSensitiveDetector is an abstract base class which represents a detector. The principal
mandate of a sensitive detector is the construction of hit objects using information from
steps along a particle track. The ProcessHits() method of G4VSensitiveDetector  performs this
task using G4Step objects as input*/
#ifndef SensitiveDetector_h
#define SensitiveDetector_h 1

#include "G4VSensitiveDetector.hh"

class G4Step;
class G4HCofThisEvent;
class G4TouchableHistory;
class G4Event;
class SensitiveDetector : public G4VSensitiveDetector
{

public:
    SensitiveDetector(G4String name);
    ~SensitiveDetector();

    /*This method is invoked at the beginning of each event. The argument of this method is an
object of the G4HCofThisEvent class. Hits collections, where hits produced in this particular
event are stored, can be associated to the G4HCofThisEvent object in this method. The hits
collections associated with theG4HCofThisEvent  object during this method can be used for
``during the event processing'' digitization.G4Event has a G4HCofThisEvent class object, that
is a container class of collections of hits. Hits collections are stored by their pointers, of
the type of the base class.*/
    void Initialize(G4HCofThisEvent*HCE);
    /*This method is invoked by G4SteppingManager when a step is composed in the
G4LogicalVolume which has the pointer to this sensitive detector. The firstargument of this
method is a G4Step  object of the current step. The second argument is a G4TouchableHistory
object for the ``Readout geometry'' describedin the next section. The second argument is NULL
for the case ``Readout geometry'' is not assigned to this sensitive detector. In this method,
one or moreG4VHit objects should be constructed if the current step is meaningful for your
detector.*/
    G4bool ProcessHits(G4Step*aStep,G4TouchableHistory*ROhist);
    /*This method is invoked at the end of each event. The argument of this method is the same
object as the previous method. Hits collections occasionally created in your sensitive
detector can be associated to the G4HCofThisEvent object.*/
    void EndOfEvent(G4HCofThisEvent*HCE);
        G4double GetDose() const;

private:
    G4double Dose;


};

#endif
```

A.3.13 SensitiveDetector.cc

```
#include "SensitiveDetector.hh"
#include "G4Step.hh"
#include "G4Event.hh"

#include "G4ParticleGun.hh"
#include "PrimaryGeneratorActionGPS.hh"


#include "G4HCofThisEvent.hh"
#include "G4TouchableHistory.hh"

#include "RunAction.hh"
#include "G4RunManager.hh"

#include <stdio.h>
#include <time.h>

#include "G4VisAttributes.hh"



SensitiveDetector::SensitiveDetector(G4String name) : G4VSensitiveDetector(name){



}

SensitiveDetector::~SensitiveDetector(){;}
```

```
/*This method is invoked at the beginning of each event. The argument of this method is an
object of the G4HCofThisEvent class. Hits collections, where hits produced in this particular
event are stored, can be associated to the G4HCofThisEvent object in this method. The hits
collections associated with the G4HCofThisEvent  object during this method can be used for
``during the event processing'' digitization.*/
void SensitiveDetector::Initialize(G4HCofThisEvent* HCE){

    Dose = 0;

}

/*This method is invoked by G4SteppingManager when a step is composed in the G4LogicalVolume
which
has the pointer to this sensitive detector. The first argument of this method is a G4Step
object of
the current step. The second argument is a G4TouchableHistory object for the ``Readout
geometry'' described
in the next section. The second argument is NULL for the case ``Readout geometry'' is not
assigned to
this sensitive detector. In this method, one or more G4VHit objects should be constructed if
the current
step is meaningful for your detector.*/
G4bool SensitiveDetector::ProcessHits(G4Step* theStep, G4TouchableHistory*){


        const G4double edep = theStep->GetTotalEnergyDeposit();



        if(edep !=0)
        {
                Dose += edep;
        }
        /*else
        {
                //G4Colour white (1.0, 1.0, 1.0);
                G4VisAttributes* TrackVisAtt = new G4VisAttributes();
                TrackVisAtt -> SetVisibility(false);
                theStep->GetTrack()->GetTouchable()->GetVolume()->GetLogicalVolume()-
>SetVisAttributes(TrackVisAtt);
        }*/

//need to return boolean variable of true
        return true;

}

/*This method is invoked at the end of each event. The argument of this method is the same
object as the previous
method. Hits collections occasionally created in your sensitive detector can be associated to
the G4HCofThisEvent object.*/
void SensitiveDetector::EndOfEvent(G4HCofThisEvent*)
{
//if the dose for this event is non zero
        if(Dose > 0 *keV)
        {
            //apparently so, so we pass the dose for this event to the method of the run
aciton that accumulates dose per event
            RunAction* myRunAction = (RunAction*)(G4RunManager::GetRunManager()-
>GetUserRunAction());

            if(myRunAction)
                myRunAction->ProcessDoseEvent(Dose);



        }
}

G4double SensitiveDetector::GetDose() const
   { return Dose; }
```

A.3.14 SteppingAction.hh

```
#ifndef SteppingAction_hh
#define SteppingAction_hh 1

#include "DetectorConstruction.hh"
#include "G4UserSteppingAction.hh"
#include "G4ThreeVector.hh"

class RunAction;

class SteppingAction : public G4UserSteppingAction
{
public:

    SteppingAction(RunAction*, DetectorConstruction* Detector);
        ~SteppingAction(){};

    void UserSteppingAction(const G4Step*);

        G4double killAngle;

private:

        G4ThreeVector ParticlePosition;
        G4ThreeVector ParticleDirection;

        G4ThreeVector DetectorPosition;
        G4ThreeVector DetectorDirection;

    RunAction* LocalRA;
        DetectorConstruction* LocalDetector;
};

#endif
```

A.3.15 SteppingAction.cc

```
#include "SteppingAction.hh"
#include "G4SteppingManager.hh"
#include "RunAction.hh"
#include "G4Step.hh"
#include "DetectorConstruction.hh"
#include "G4ThreeVector.hh"

//for sensitive detector infomation
#include "SensitiveDetector.hh"
#include "G4SDManager.hh"

#include "G4RunManager.hh"

#define PI 3.14159265

SteppingAction::SteppingAction(RunAction* RA, DetectorConstruction* Detector):LocalRA(RA),
LocalDetector(Detector){

        killAngle = 30;

}

void SteppingAction::UserSteppingAction(const G4Step* TheStep)
{

        if (LocalDetector->killZone == true)
        if(TheStep ->GetPostStepPoint()->GetPhysicalVolume() != 0){

                G4String CurrVol = TheStep->GetPostStepPoint()->GetPhysicalVolume()-
>GetName();

                /*if (CurrVol == "ground_phys"){
                        G4String ParName = (TheStep->GetTrack()->GetDynamicParticle()-
>GetDefinition()->GetParticleName());
                        if (ParName != "gamma"){
                                TheStep->GetTrack()->SetTrackStatus(fKillTrackAndSecondaries);
                                //G4cout << "A " << ParName << " track killed below ground" <<
G4endl;
                        }
                }
                else */

                if (CurrVol == "KillZone_phys")
                {
                        G4String ParName = (TheStep->GetTrack()->GetDynamicParticle()-
>GetDefinition()->GetParticleName());

                        ParticlePosition = TheStep->GetPostStepPoint()->GetPosition();
                        ParticleDirection = TheStep->GetPostStepPoint()-
>GetMomentumDirection();       // This is a unit vector

                        DetectorPosition = LocalDetector->GetPosition();

                        DetectorDirection = DetectorPosition - ParticlePosition;     //This
actually seems to work.
                        //G4cout << "Detector direction is (" << DetectorDirection[0]/m << ","
<< DetectorDirection[1]/m << "," << DetectorDirection[2]/m << ")" << G4endl;

                        double DetectorDirectionMag =
sqrt(DetectorDirection[0]*DetectorDirection[0] + DetectorDirection[1]*DetectorDirection[1] +
DetectorDirection[2]*DetectorDirection[2]);

                        DetectorDirection[0] = DetectorDirection[0]/DetectorDirectionMag;
                        DetectorDirection[1] = DetectorDirection[1]/DetectorDirectionMag;
                        DetectorDirection[2] = DetectorDirection[2]/DetectorDirectionMag;

                        // double dotProduct = DetectorDirection[0]*ParticleDirection[0] +
DetectorDirection[1]*ParticleDirection[1] + DetectorDirection[2]*ParticleDirection[2];
                        //G4cout << "Dot Product is " << dotProduct << G4endl;

                        double DirectionDifference =
acos(DetectorDirection[0]*ParticleDirection[0] + DetectorDirection[1]*ParticleDirection[1] +
DetectorDirection[2]*ParticleDirection[2])*180/PI;
                        //G4cout << "Direction Difference is " << DirectionDifference <<
G4endl;

                        killAngle = LocalDetector->GetKillZoneAngle();

                        if (DirectionDifference > killAngle)
                        {
                                //TheStep->GetTrack()-
>SetTrackStatus(fKillTrackAndSecondaries);
                                TheStep->GetTrack()->SetTrackStatus(fStopAndKill);
                        }
                        //G4cout << "A " << ParName << " track killed in the kill zone" <<
G4endl;
                }


        }



/* Only required if analysis intended to use ROOT
#ifdef G4ANALYSIS_USE_ROOT
        if (OldVolName == "WaterBox1_phys"||OldVolName == "BoxP")
        //if (OldVolName == "WaterBox1_phys")
        {


                if(CurrVol == "scoring_phys")
                {
                    G4String ParName = (TheStep->GetTrack()->GetDynamicParticle()-
>GetDefinition()->GetParticleName());
```

```
                       G4double En = (TheStep->GetTrack()->GetKineticEnergy());
                       G4int ID = (TheStep->GetTrack()->GetTrackID());


                       //G4cout << "A " << ParName << " entered the Sensitive Detector with
kinetic energy " << En /MeV << " MeV" << G4endl;

                       LocalRA->ProcessStep(En);

               }
            }


#endif
*/

}
```

A.4. Example Macro Files

A.4.1 BuildEnvironment.mac

This is an example Environment macro file. “#” is a comment character.

```
/USER/Block/RemoveAll

/USER/Block/Size 20 5 0.1 m
/USER/Block/Material  G4_CONCRETE
/USER/Block/Position 20 5 1 m
/USER/Block/Build

#/USER/Block/Size 20 5 0.1 m
#/USER/Block/Material  G4_CONCRETE
#/USER/Block/Position 20 5 -1 m
#/USER/Block/Build


#/USER/Block/Size 1 1 0.1 m
#/USER/Block/Material  G4_CONCRETE

#/USER/Block/Position 2 1 1 m
#/USER/Block/Build

#/USER/Block/Position -2 1 1 m
#/USER/Block/Build

#/USER/Block/Position 6 1 1 m
#/USER/Block/Build

#/USER/Block/Position -6 1 1 m
#/USER/Block/Build

#/USER/Block/Position 10 1 1 m
#/USER/Block/Build

#/USER/Block/Position -10 1 1 m
#/USER/Block/Build

#/USER/Block/Position 14 1 1 m
#/USER/Block/Build

#/USER/Block/Position -14 1 1 m
#/USER/Block/Build

# Search Test
#/USER/Block/Size 2 2 0.05 m
#/USER/Block/Position 50 2 48 m
```

```
#/USER/Block/Material G4_CONCRETE
#/USER/Block/Build

#/USER/Block/Size 4 2 0.05 m
#/USER/Block/Position 0 2 -1.5 m
#/USER/Block/Material G4_CONCRETE
#/USER/Block/Build

#/USER/Block/Size 0.5 2 0.01 m
#/USER/Block/Position -4.5 2 -1.5 m
#/USER/Block/Material G4_GLASS_PLATE
#/USER/Block/Build

#/USER/Block/Size 0.05 2 1.5 m
#/USER/Block/Position -5 2 0 m
#/USER/Block/Material  G4_CONCRETE
#/USER/Block/Build
#Search Test

#/USER/Block/Size 1 1 0.1 m
#/USER/Block/Material G4_Pb
#/USER/Block/Position 2 1 2 m
#/USER/Block/Build

#/USER/Block/Size 0.05 1 1 m
#/USER/Block/Position 1 1 0 m
#/USER/Block/Material G4_CONCRETE
#/USER/Block/Build


#/USER/Block/Size 0.05 5 5 m
#/USER/Block/Position 5 5 0 m
#/USER/Block/Material  G4_CONCRETE
#/USER/Block/Build

#/USER/Block/Size 5 5 5 m
#/USER/Block/Position 10 5 0 m
#/USER/Block/Material  G4_CONCRETE
#/USER/Block/Build

#/USER/Block/Size 0.25 5 5 m
#/USER/Block/Position 2 5 0 m
#/USER/Block/Material  G4_Pb
#/USER/Block/Build
```

A.4.2 vis_vrml.mac

This is an example VRML visualisation macro. "#" is a comment character.

```
/vis/scene/create
#/vis/open OGLSWin32
/vis/open VRML2FILE
#/vis/viewer/reset

/tracking/storeTrajectory 0
/vis/scene/add/trajectories
/vis/scene/add/hits
/vis/viewer/set/style solid
#/vis/viewer/set/viewpointThetaPhi 0 0
/vis/viewer/set/lightsThetaPhi 45 45
#/vis/viewer/set/targetPoint 0 0 0 m
#/vis/viewer/zoom 50
#/vis/drawVolume
#/vis/viewer/update

#Draws particles of different energy levals in different colours
#/vis/modeling/trajectories/create/drawByAttribute
#/vis/modeling/trajectories/drawByAttribute-0/setAttribute IMag
```

```
#/vis/modeling/trajectories/drawByAttribute-0/addInterval interval1 0.0 keV 100.0 keV
#/vis/modeling/trajectories/drawByAttribute-0/addInterval interval2 100.0 keV 300.0 keV
#/vis/modeling/trajectories/drawByAttribute-0/addInterval interval3 300.0 keV 600.0 keV
#/vis/modeling/trajectories/drawByAttribute-0/addInterval interval4 600.0 keV 800.0 keV
#/vis/modeling/trajectories/drawByAttribute-0/addInterval interval5 800.0 keV 2000.0 keV
#/vis/modeling/trajectories/drawByAttribute-0/interval1/setLineColourRGBA 1 0.5 0 1
#/vis/modeling/trajectories/drawByAttribute-0/interval2/setLineColourRGBA 1 1 0 1
#/vis/modeling/trajectories/drawByAttribute-0/interval3/setLineColourRGBA 0 1 0 1
#/vis/modeling/trajectories/drawByAttribute-0/interval4/setLineColourRGBA 0 1 1 1
#/vis/modeling/trajectories/drawByAttribute-0/interval5/setLineColourRGBA 0 0 1 1

#Only Draws the following particles
/vis/filtering/trajectories/create/particleFilter
/vis/filtering/trajectories/particleFilter-0/add proton
/vis/filtering/trajectories/particleFilter-0/add e+
/vis/filtering/trajectories/particleFilter-0/add e-
/vis/filtering/trajectories/particleFilter-0/add ion
/vis/filtering/trajectories/particleFilter-0/add gamma
/vis/filtering/trajectories/particleFilter-0/add neutron


#accumulate the events at end of each event, ie overlay trajectories max 500
/vis/scene/endOfEventAction accumulate 500
```

A.4.3 matlab_generated.mac

This is an example macro file that is generated from MATLAB each time GEANT4 is called. This file is generated by the function GetCountsG4.m, see A.2.2.

```
/control/execute BuildEnvironment.mac
/gps/pos/type Point
/gps/particle gamma
/gps/energy 1.3 MeV
/gps/ang/type iso
/gps/pos/centre -3 1 0 m
/USER/DetectorPosition 0 1 -94 m
/USER/KillZone/set true
/USER/KillZone/Radius 3
/USER/KillZone/Angle 30
/USER/update
/run/beamOn 1
/USER/update
/run/beamOn 792000
```

DEFENCE SCIENCE AND TECHNOLOGY ORGANISATION DOCUMENT CONTROL DATA	1. PRIVACY MARKING/CAVEAT (OF DOCUMENT)
2. TITLE A Proof-of-concept Study on Integrating GEANT4 and MATLAB to Develop a Simulation Capability for Testing Radiological Source Localisation Algorithms	3. SECURITY CLASSIFICATION (FOR UNCLASSIFIED REPORTS THAT ARE LIMITED RELEASE USE (L) NEXT TO DOCUMENT CLASSIFICATION) Document (U) Title (U) Abstract (U)
4. AUTHOR(S) Alaster Meehan, Ajith Gunatilaka, Damian Marinaro and Michael Roberts	5. CORPORATE AUTHOR DSTO Defence Science and Technology Organisation 506 Lorimer St Fishermans Bend Victoria 3207 Australia

6a. DSTO NUMBER DSTO-TN-0995	6b. AR NUMBER AR-014-956	6c. TYPE OF REPORT Technical Note	7. DOCUMENT DATE February 2011

8. FILE NUMBER 2010/1148441	9. TASK NUMBER 07/335	10. TASK SPONSOR VCDF	11. NO. OF PAGES 59	12. NO. OF REFERENCES 7

DSTO Publications Repository http://dspace.dsto.defence.gov.au/dspace/DSTO-TN-0995	14. RELEASE AUTHORITY Chief, Human Protection and Performance Division

15. SECONDARY RELEASE STATEMENT OF THIS DOCUMENT

Approved for public release

OVERSEAS ENQUIRIES OUTSIDE STATED LIMITATIONS SHOULD BE REFERRED THROUGH DOCUMENT EXCHANGE, PO BOX 1500, EDINBURGH, SA 5111

16. DELIBERATE ANNOUNCEMENT

No Limitations

17. CITATION IN OTHER DOCUMENTS Yes

18. DSTO RESEARCH LIBRARY THESAURUS

GEANT4, MATLAB, radiological simulation, source backtracking

19. ABSTRACT

This report describes a proof-of-concept study on integrating a Monte Carlo particle simulation package called GEANT4 with MATLAB technical computing software to build a radiological simulation capability to test and evaluate radiological source localisation algorithms developed in MATLAB. The purpose of this report is to document what was learnt during this project, including alternative approaches that were considered; useful instructions and tips; and bugs and pitfalls, so that these can benefit anyone who is undertaking to extend this work.